Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 379 (2007) 84–119
Theoretical Computer Science
www.elsevier.com/locate/tcs

Strong planning under uncertainty in domains with numerous but identical elements (a generic approach)

Max Kanovich[a,*], Jacqueline Vauzeilles[b]

[a] *Computer Science Department, Queen Mary, University of London, Mile End Road, London, E1 4NS, UK*
[b] *LIPN, UMR CNRS 7030, Institut Galilée, 99 Av. J.-B. Clément, 93430 Villetaneuse, France*

Received 11 April 2005; received in revised form 25 January 2007; accepted 28 January 2007

Communicated by M.W. Mislove

Abstract

The typical AI problem is that of making a plan of the actions to be performed by a robot so that the robot could get into a set of final situations, if it started with a certain initial situation. The planning problem is known to be generally very complex. Even within the case of 'well-balanced' actions, strong planning under uncertainty about the effects of actions, or games such as 'Robot against Nature', is EXPTIME-complete. As a result, AI planners are very sensitive to the number of the variables involved in making a plan, the inherent symmetry of the problem, and the nature of the logical formalisms being used.

This paper shows that linear logic provides a convenient and adequate tool for representing strong and weak planning problems in non-deterministic domains.

A particular focus of this paper is on planning problems with an unbounded number of functionally identical objects. We show that for such problems linear logic is especially effective and leads to a dramatic contraction of the search space from exponential to polynomial in size.

We employ the ability of linear logic to reason about multisets, which in this instance are created by identifying several distinct objects as being functionally equivalent for the problem at hand (think of a number of balls, each of which must be moved to some new location — the balls are distinct, but are functionally equivalent for the problem).

In linear logic terms, we establish a clear syntactic condition that allows us to show that solving a generic planning problem where there is only one generic object, directly implies a solution to the original real planning problem over several real objects, the isomorphic copies of the generic object.

Moreover, this correspondence also guarantees to produce a real solution that works in polynomial time.

Keywords: AI planning under uncertainty; Linear logic; Proofs as programs; Deterministic and non-deterministic planning domains; Horn linear logic

1. Introduction and motivating examples

The aim of this paper is to show that linear logic can automatically exploit peculiarities of some AI systems, and achieve a significant speedup over traditional approaches by decreasing the combinatorial costs associated with searching large spaces.

* Corresponding author. Tel.: +44 2078825213.
E-mail addresses: mik@dcs.qmul.ac.uk (M. Kanovich), jv@lipn.univ-paris13.fr (J. Vauzeilles).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2007.01.022

The AI systems we consider have the following features:

A *robot* is dealing with a finite number of *objects*. Each of the *actions* performed by the robot results in correlations *newly established* between the objects, with some old correlations being *destroyed*. The conditions *enabling* an action includes the presence of certain old correlations.

The typical AI problem is that of *making a plan* of the actions to be performed by the robot so that it could get into a set of *final situations* $\widetilde{Z}$, if it started with a certain *initial situation* W (see, for instance, [30,3,28,11,24,22,5]).

In addition to [30,3,11], we consider here *two kinds of non-determinism* related to the points from which the robot activity could be regarded [28,22,5]:

(1) First, the robot makes its *own choice* between the actions from a given set.
(2) Second, the effect of the action chosen may be *non-deterministic* because the robot does not know, *in advance*, what the reaction of Nature would be under the particular circumstances.

We will address the following versions of the planning problem in *non-deterministic domains* under *uncertainty* about the effects of the actions [5,8]:

(a) Strong planning, in which the planning objective is to find a plan that is guaranteed to achieve the goal even within the "worst-case scenario".
(b) Weak planning, in which the planning objective is to find an "optimistic" plan that has non-zero probability of achieving the goal.

There are a number of logical formalisms for handling the above AI planning problem (see, for instance, [30,3,28, 11,24,22,5]).

As a logical formalism to specify and sort out planning problems under uncertainty, we use *linear logic* introduced by Girard [16] as a resource-sensitive refinement of traditional logic. Linear logic provides a convenient and adequate tool for sorting out planning problems in deterministic as well as in non-deterministic domains [28,29,22]. The fundamental advantage of linear logic is that it yields a direct and transparent correspondence between proofs for Horn linear logic sequents and plans for AI problems [28,29,19,22].

Along the way, a linear logic formalism allows us to establish the complexity bounds for the strong planning problem under uncertainty, as well as for the weak planning problem under uncertainty.

Notice that even the simple existence problem (that is whether there is a strong plan for the above task $W \Rightarrow \widetilde{Z}$) is PSPACE-complete for propositional deterministic domains [6], and EXPTIME-complete for propositional non-deterministic domains [22,27,18].

But a simple decision procedure is not satisfactory for our purposes: planners should sort out the much more complicated problem of making an *actual plan*. The effect is that the planners are much more sensitive to the number of the variables involved in making plans, functional similarity among objects, and the nature of the logical formalism being used.

In particular, *computer-aided* planners run into difficulties caused by the combinatorial costs associated with searching large spaces for the following planning problems in AIPS Planning Competitions [1,2] and the like (whereas their *common sense* solutions are obvious from the human point of view!).

Example 1.1. *"Gripper"* [1]: There is a robot with two grippers. It can carry a ball in each. The goal is to take N balls from one room to another.[1]

Example 1.2. *"Elevator"* [2]: There is an elevator that allows only 6 people to ride at a time. The goal is to move N passengers to their destination.

Example 1.3. *"Pigeonhole Principle"*: The goal is to put N balls into k containers so that there is no container with more than one ball.

[1] An excerpt from [1]: "STRIPS representation leads to a combinatorial explosion in the search space. All planners obviously suffer from this explosion, with the exception of HSP that does quite well. Interestingly, HSP plans only to transport one ball at a time leading to lots of unnecessary move actions .... IPP has been run with RIFO switched on, which excludes one gripper as irrelevant, i.e. non-optimal plans are found using only the left gripper.... IPP and BLACKBOX don't even provide data on the harder problem instances".

As a matter of fact, the combinatorial explosion in the search spaces related to these examples stems primarily from the fact that the domains under consideration contain a large number of objects, which gives rise to a exponentially large number of reachable configurations.

On the other hand, the domains are highly symmetrical, since the balls, grippers, people and containers in question are supposed to be *identical*, *indistinguishable*, *interchangeable*, etc.

Our ambition here is, by means of a specific *erasing individuality* method, to turn this symmetry feature to our favour and to obtain polynomial solutions in polynomial time.

Example 1.4. *"Towers of Hanoi"*. A one-handed robot deals with N plates $a_1, .., a_n, a_{n+1}, .., a_N$ of increasing size, and three boxes B_1, B_2, B_3.

Initially, the plates are stacked inside B_1 in the natural order, the largest plate being the bottom one.

The goal is to move the first n plates from B_1 to B_2, one plate at a time, in such a way that no plate is ever placed on a smaller one. Box B_3 may be used for temporary storage of plates.

The reason for showing Example 1.4 here is that it has no symmetry among the plates, so we could not have relied upon symmetry for the production of *polynomial plans* promised for the previous examples. And this is in full accordance with the fact that every plan for Example 1.4 has exponential length $\Omega(2^n)$ at least. ■

Within the above examples, the effect of each of the actions involved happened to be *deterministic*.

The situation becomes much more complicated in the case of the *strong planning under uncertainty about the effects of actions*. Within this non-deterministic paradigm, the strong planning objective is to find a plan that is guaranteed to achieve the goal even within the "worst-case scenario". In particular, we will consider *knowledge acquisition* games where a plan we have to look for becomes a *winning strategy* against Nature, as in the following folklore examples.

Example 1.5. *"Socks"*: You have a drawer full of socks, N red socks (all identical) and N blue socks. The challenge is to pick out a minimal number of socks to assure that you have at least one matching pair.

Example 1.6. *"Omelette"*: We assume to have N eggs that can be grabbed and broken into a bowl. Eggs can be unpredictably good or bad. It is possible to determine whether an egg is good or bad only after the egg is broken into the bowl. The rotten egg in the bowl has the effect of spoiling the bowl, but the bowl can always be cleaned by discarding its contents.

The intended goal is to have n good eggs in the bowl, so that an omelette can be prepared. (The case where $N=\infty$ and $n=2$ is discussed in [8].)

We observe the combinatorial explosion in the search spaces within these examples, as well: the real search spaces contain $\Omega(2^N)$ configurations.

But, in addition to the previous deterministic domains, there is another trouble-making dimension here: the strong plans we have to look for will be of hyper-exponential size $\Omega(2^{2^N})$, whenever we take these plans in the form of tree-like winning strategies over these real search spaces exploded.

In spite of this discouraging fact, we will show how to use the symmetry caused by the indistinguishability of socks and eggs within the above non-deterministic domains in order to *circumvent* the combinatorial explosion in their real search spaces and produce plans (in a folded form) in polynomial time. ■

Our paper focuses on strong and weak planning problems in non-deterministic domains involving an unbounded number of functionally identical objects. To address the key issue: *'How to recognize functional similarity among objects and break the combinatorial explosion caused by the large number of identical objects',* we show that linear logic can radically reduce the number of variables involved in making plans and provide thereby *polynomial* solutions to such planning problems.

The main idea of our present approach is the following:

(1) First, having detected symmetry within a given system, we break it by replacing the *unbounded number* of specific names of objects with *one* 'generic' name, so that we can solve a *mock* 'generic' problem with a drastically smaller state space (polynomial instead of exponential) **but** over the 'generic' object.

(2) Second, each of the *mock* solutions dealing with one 'generic' object is proved to be *directly* translatable into an (optimal) polytime solution to the original *real* planning problem dealing with the unbounded number of 'real' objects.

In other words, for a problem in which N objects, say $b_1, b_2,.., b_N$, *cannot be individualized* within the system,

(1) First, we intend to treat them as *identical copies* of a single 'generic' object, say b, with the *set* $\{b_1, b_2, .., b_N\}$ being replaced with the *multiset* $\{b, b, .., b\}$.
This is expected to provide a *polynomial* search space but over one 'generic' b.
(2) Second, having found a solution to the 'generic' planning problem, we have to convert it (in polytime) into a real solution to the original real problem.

The basic problems to be sorted out within our approach are as follows:

(a) Detect symmetry within a given system related to functional similarity among certain objects.
This can be done by analysis of the actions allowed and the format of inputs and outcomes.
(b) Reformulate a real problem as a 'generic' one.
This can be easily done by 'erasing indices'.
(c) Find 'generic' plans (or winning strategies) over the 'generic' configurations.
This can be done by means of a number of techniques including direct search for the shortest paths, theorem proving, BDDs, generalized BDDs, SAT, Petri nets, game theory, linear logic programming, etc.
(d) Convert the 'generic' plan found into a real plan over the real configurations.
This is readily seen to be applicable only for a restricted class of systems. Here we give a transparent syntactic condition (see Definition 4.4) that guarantees our approach work properly for a very wide class of problems (including the above examples and the like).
(e) Provide the feasibility of each of the above steps.
(e1) The syntactic condition introduced in Definition 4.4 allows us to detect the symmetry we can use for making plans of polynomial size. (See, for instance, Example 4.1 and Comment 4.4)
(e2) Reformulation of the real problem as a 'generic' one is easy.
(e3) Generally, the 'generic' search space happens to be small, so that 'generic' plans can be found, if they exist, in polynomial time.
(e4) Based on our syntactic condition, we will show how any 'generic' plan can be easily transformed into a real plan over the real configurations.

1.1. Summary of the paper: When and why it works

The present paper is a follow-up to the papers that started linear logic applications to AI [28,22]: in [28] AI planning problems are encoded and solved within linear logic; in [22] the complexity of planning problems encoded in linear logic is studied. The present paper is based on our conference paper [23].

To make this paper more easily comprehensible, in Section 1.2 we give a brief review of linear logic, and in Section 1.3 we give basic details about the tool STRIPS that is currently used in solving AI planning problems.

Closing the introductory part, in Section 1.4 we illustrate in details how our 'generic' approach works for *deterministic* domains.

In Section 2 we are listing the basic features of the linear logic paradigm that support our approach to planning problems. In addition, as a working fragment of linear logic for the planning problems, we introduce *Horn linear logic*.

In Section 3 we introduce our central concept of strong and weak solutions to planning problems under uncertainty caused by actions with non-deterministic effects.

In Section 4 we prove Theorem 4.1 that justifies our 'generic' approach. Among other things, we establish a transparent *correspondence* between strong solutions to planning problems under uncertainty and linear logic proofs for the corresponding Horn sequents.

The complexity issues are considered in Section 5.

In Section 6 we show how to extend our results to the weak planning problems under uncertainty.

1.2. Linear logic syntax: Background

Let us recall the background material on linear logic with which we are dealing in this paper (see, for instance, Girard [16]).

In this section we give a short sketch of syntactic peculiarities of linear logic; we will address interpretation issues in Section 2.

1.2.1. Connectives

Within linear logic,

(i) the traditional conjunction $\wedge$ is split into two connectives: $\otimes$ (*tensor*) and & (*with*);
(ii) dually, the traditional disjunction $\vee$ is split into two connectives: $\parr$ (*par*) and $\oplus$ (*plus*).

The so-called 'multiplicative' connective $\otimes$ is governed by the following inference rules:

$$\mathbf{L}\otimes\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \otimes B) \vdash \Delta} \qquad \mathbf{R}\otimes\ \frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2 \vdash (A \otimes B), \Delta_1, \Delta_2}$$

while the 'additive' connective & is governed by the following inference rules:

$$\mathbf{L}\&_1\ \frac{\Gamma, A \vdash \Delta}{\Gamma, (A\&B) \vdash \Delta} \quad \mathbf{L}\&_2\ \frac{\Gamma, B \vdash \Delta}{\Gamma, (A\&B) \vdash \Delta} \qquad \mathbf{R}\&\ \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A\&B), \Delta}$$

The rules for $\parr$ and $\oplus$ are defined in a straightforward dual manner.
In particular, for $\oplus$ we have the following:

$$\mathbf{L}\oplus\ \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \oplus B) \vdash \Delta} \qquad \mathbf{R}\oplus_1\ \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash (A \oplus B), \Delta} \quad \mathbf{R}\oplus_2\ \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash (A \oplus B), \Delta}$$

Linear implication $(A \multimap B)$ is defined as $(A^{\perp} \parr B)$, with the following rules:

$$\mathbf{L}\multimap\ \frac{\Gamma_1 \vdash \Delta_1, A \quad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, (A \multimap B), \Gamma_2 \vdash \Delta_1, \Delta_2} \qquad \mathbf{R}\multimap\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash (A \multimap B), \Delta}$$

1.2.2. Structural rules

As for structural rules such as Permutation, Contraction and Weakening, only Permutation Rule is assumed within linear logic:

$$\frac{\Gamma_1,\ A, B,\ \Gamma_2 \vdash \Delta}{\Gamma_1,\ B, A,\ \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta_1,\ A, B,\ \Delta_2}{\Gamma \vdash \Delta_1,\ B, A,\ \Delta_2}$$

Cut Rule

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

can be eliminated in linear logic.
As compared to traditional logics, the fundamental formal difference is that the Contraction Rule

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

is not assumed in linear logic.
Linear logic enlarged with the Weakening Rule

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}$$

is called *affine logic*.

Example 1.7. The absence of Contraction provides that, for instance, $p \vdash (p \otimes p)$ is not provable even in affine logic.

1.2.3. Horn-like fragments of linear logic

Following [19], we confine ourselves to simple fragments of linear logic that operate with *pure Horn sequents* of the form

$$X \vdash Y$$

and *disjunctive Horn sequents* of the form

$$X \vdash (Y_1 \oplus \cdots \oplus Y_m),$$

where X and Y's are built up from atomic predicate expressions $Q(t_1, .., t_k)$ only by $\otimes$.

Though Horn-like fragments of linear logic are the simplest ones from the syntactical point of view (we need neither explicit negation $A^{\perp}$, nor $\parr$, nor embedded implications), linear logic proof machinery provides the extreme expressive power of the Horn-like fragments of linear logic (see Section 2).

1.3. AI planning: Backgrounds

Here we recap the background material on AI planning, following Nilsson's STRIPS [30,11], which is the base for most of the languages for expressing automated planning problem instances.

Definition 1.1. Given an AI system with pure deterministic actions,

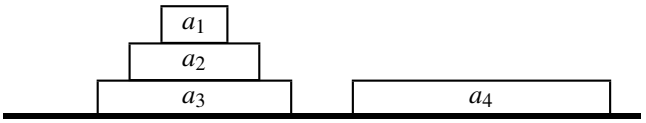
(a) A *situation*, or a *state*, within, the AI system is described as a set of atomic predicate formulas and/or their negations that are true in it;
(b) Each action α is specified in terms of its *precondition* $Pre(\alpha)$, which consists of atomic predicate formulas and/or their negations, and two lists of atomic predicate formulas: *add-list* $Add(\alpha)$ and *delete-list* $Del(\alpha)$;
(c) The action specification is applied to *edit* descriptions of situations instead of being used as an *axiom* in *deducing properties* of situations.

Namely, if α's precondition $Pre(\alpha)$ is met, generating a new situation description from an old one is a matter of deleting all the atomic formulas taken from $Del(\alpha)$ and adding all the atomic formulas taken from $Add(\alpha)$.

Example 1.8. Within *"Towers of Hanoi"* in Example 1.4, action $\mathtt{move}_{124}$ — that the robot makes to take plate a_1 stacked on plate a_2 and place a_1 on plate a_4, can be specified in STRIPS as:

(a) $Pre(\mathtt{move}_{124}) = \{\mathtt{clear}(a_1), \mathtt{on}(a_1, a_2), \mathtt{clear}(a_4)\}$,
(b) $Del(\mathtt{move}_{124}) = \{\mathtt{on}(a_1, a_2), \mathtt{clear}(a_4)\}$,
(c) $Add(\mathtt{move}_{124}) = \{\mathtt{clear}(a_2), \mathtt{on}(a_1, a_4)\}$.

Here $\mathtt{on}(x, y)$ means *"plate x is on plate y"*, and $\mathtt{clear}(x)$ stands for *"no plate is on plate x"*.
Accordingly, applying this 'deletion–addition' machinery of action $\mathtt{move}_{124}$ to a state of the form



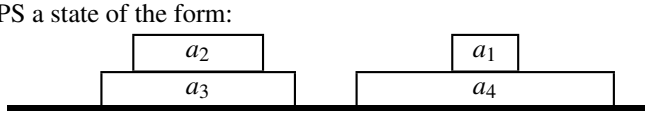
described in STRIPS by the set

$$\{\mathtt{clear}(a_1), \mathtt{on}(a_1, a_2), \mathtt{on}(a_2, a_3), \mathtt{clear}(a_4)\}, \tag{1}$$

we result in the following set:

$$\{\mathtt{clear}(a_2), \mathtt{on}(a_2, a_3), \mathtt{clear}(a_1), \mathtt{on}(a_1, a_4)\}, \tag{2}$$

which represents in STRIPS a state of the form:



■

Definition 1.2. Given an initial state W and a goal state Z, a *plan* for such a STRIPS planning task:

$$W \Rightarrow Z,$$

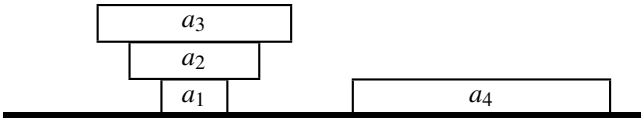
is a linear-ordered sequence of actions that can be executed from W and that leads to Z.

Informally, a plan $\mathcal{P}$ is defined as a chain of *commands*, or *moves*, to perform certain actions [30,3,28], something like:

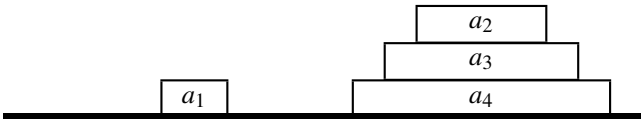
(i) apply α_{i_1};
(ii) apply α_{i_2};
(iii) .....
(iv) apply α_{i_n};
(v) halt;

Example 1.9 (*Continuing Example* 1.8). By $\texttt{move}_{ijk}$ we denote the action that the robot makes to take plate a_i stacked on plate a_j and place a_i on plate a_k.

Let W be an *initial state* of the form



and let Z be a *goal state* of the form



Then the following plan leads from W to Z:

(i) $\texttt{move}_{324}$;
(ii) $\texttt{move}_{213}$;
(iii) halt; ■

The AI planning problem is known to be generally very complex.

(a) Even for a propositional STRIPS, deciding the simple existence of a plan is PSPACE-complete [6].
(b) As for making plans themselves, we run into additional difficulties with the fact that certain examples require plans of exponential length (see Example 1.4).

1.3.1. Linear logic versus STRIPS

The above complexity results are in accordance with the following obstructions to resolving AI planning problems in *purely logical terms* discussed in the literature — that within the traditional logical paradigm we can express only STRIPS without 'deletion' effects (in fact, the latter corresponds to a Horn fragment of Boolean logic). In order to cope with 'deletion' effects, STRIPS [30,11] was invented as a non-logical system in which an action specification was applied to *edit* descriptions of situations instead of being used as an axiom in *deducing* properties of situations.

But later it was shown that the whole of STRIPS, with 'deletion' effects included, received an adequate and comprehensive *purely logical* representation within linear logic [28,22].

The main idea of the logical approach in [28,22] is as follows:

(a) *do not* change the "naive" Horn-like specification language,
(b) but *refine* the system of logical inference rules as linear logic rules, with the result that AI systems can be properly handled in purely logical terms.

In fact, [28,22] invoke a syntactically simple fragment of linear logic comprised of Horn-like formulas, providing the desired 'deletion–addition' effects by linear logic proof machinery (see also Section 2).

Example 1.10. According to [28,22], action $\mathtt{move}_{124}$ from Example 1.8 is axiomatized as the following pure Horn sequent[2]:

$$(\mathtt{clear}(a_1) \otimes \mathtt{on}(a_1, a_2) \otimes \mathtt{clear}(a_4)) \vdash (\mathtt{clear}(a_1) \otimes \mathtt{clear}(a_2) \otimes \mathtt{on}(a_1, a_4)). \tag{3}$$

Accordingly, state (1) is encoded in linear logic as an 'elementary product' of the form

$$(\mathtt{clear}(a_1) \otimes \mathtt{on}(a_1, a_2) \otimes \mathtt{on}(a_2, a_3) \otimes \mathtt{clear}(a_4)),$$

and state (2) is encoded as

$$(\mathtt{clear}(a_2) \otimes \mathtt{on}(a_2, a_3) \otimes \mathtt{clear}(a_1) \otimes \mathtt{on}(a_1, a_4)).$$

The act of producing state (2) from state (1) by means of action $\mathtt{move}_{124}$ is expressed by the fact that the corresponding sequent

$$(\mathtt{clear}(a_1) \otimes \mathtt{on}(a_1, a_2) \otimes \mathtt{on}(a_2, a_3) \otimes \mathtt{clear}(a_4)) \vdash (\mathtt{clear}(a_2) \otimes \mathtt{on}(a_2, a_3) \otimes \mathtt{clear}(a_1) \otimes \mathtt{on}(a_1, a_4))$$

is derived from sequent (3) in linear logic.

1.4. Our generic approach within purely deterministic domains

We illustrate our approach with Example 1.11, which combines basic features of the *deterministic* domains from the above combinatorially exploded Examples 1.1–1.3.

Example 1.11. *"Briareus"*[3]: a robot has k grips. It can carry a ball in each. The goal is to take N balls from one room to another.

The number of configurations to be investigated in the planning process seems to be at least *exponential* $\Omega(2^N)$, since each of the balls, say b_1,b_2,..,b_N, has at least two independent states: "in room 1", or "in room 2".

(a) But this combinatorial explosion stems primarily from the fact that we are dealing with the *set* $\{b_1, b_2, .., b_N\}$ of N *distinct* names.
(b) Whereas the balls are supposed to be *identical*, and the initial and final configurations are *symmetrical* with respect to the balls' individual names.

Idea: Deal with the *multiset* $\{b, b, .., b\}$ consisting of N copies of one 'generic' ball, say b. (Bear in mind that, because of commutativity of multisets, the number of the corresponding 'generic configurations' is expected to be *polynomial* at the most)

Another source of a combinatorial explosion $\Omega(2^k)$ here is the unbounded number of grips, say h_1,h_2,..,h_k.

Idea: Since the grips are also *indistinguishable* and *interchangeable* within the system, the collection of all grips could be thought of as the *multiset* $\{h, h, .., h\}$ consisting of k copies of one 'generic' grip h.

To be more formal, let

(a) $\mathtt{Room}(x)$ mean *"the robot is in room x"*,
(b) $\mathtt{Hold}(y, z)$ stand for *"grip y holds ball z"*,
(c) $\mathtt{Empty}(y)$ mean *"grip y of the robot is empty"*, and
(d) $\mathtt{Floor}(x, z)$ stand for *"ball z is on the floor of room x"*.

Here x is a variable of *sort* 'room', y is of *sort* 'grip', z is of *sort* 'ball'.
The 'pick up' action $\mathtt{pick}(x, y, z)$:

[2] Here $A \otimes B$ is conceived of as "A and B co-exist together". See formalities in Section 2.

[3] During the battle against the Titans, *Briareus*, a hundred-handed giant, took advantage of his one hundred hands by throwing rocks at the Titans. Indeed, Briareus would have failed if he had wasted his time to *individualize* the rocks and hands!

"Being in room x and having the empty grip y, grasp ball z with y",

is formalized by the following

$$(\mathtt{Room}(x) \otimes \mathtt{Empty}(y) \otimes \mathtt{Floor}(x, z)) \vdash (\mathtt{Room}(x) \otimes \mathtt{Hold}(y, z)) \tag{4}$$

The 'put down' action $\mathtt{put}(x, y, z)$:

"Being in room x and holding ball z in grip y, put z down on the floor, and leave y empty",

is specified as

$$(\mathtt{Room}(x) \otimes \mathtt{Hold}(y, z)) \vdash (\mathtt{Room}(x) \otimes \mathtt{Empty}(y) \otimes \mathtt{Floor}(x, z)) \tag{5}$$

The 'move' action $\mathtt{move}(x_1, x_2)$:

"Move from room x_1 to room x_2",

is axiomatized as

$$\mathtt{Room}(x_1) \vdash \mathtt{Room}(x_2). \tag{6}$$

Within the original planning problem in Example 1.11:

$$\mathtt{In}_{N,k}(h_1, h_2, .., h_k, b_1, b_2, .., b_N) \Rightarrow \mathtt{Goal}_N(b_1, b_2, .., b_N), \tag{7}$$

we look for a *plan* leading from the initial situation

"The robot is in room 1*, its grips h_1,h_2,..,h_k are empty, and N balls b_1,b_2,..,b_N are on the floor of room* 1*"*:

formally represented as:

$$\mathtt{In}_{N,k}(h_1, .., h_k, b_1, .., b_N) = \left(\mathtt{Room}(1) \otimes \bigotimes_{j=1}^{k} \mathtt{Empty}(h_j) \otimes \bigotimes_{i=1}^{N} \mathtt{Floor}(1, b_i)\right), \tag{8}$$

into a situation where

"All N balls are in room 2*"*

or, formally,

$$\mathtt{Goal}_N(b_1, .., b_N) := \bigotimes_{i=1}^{N} \mathtt{Floor}(2, b_i) \tag{9}$$

According to our *erasing individuality* approach, we first *abstract* this original real problem (7) with the following 'generic' planning problem

$$\mathtt{In}_{N,k}(h, h.., h, b, b, .., b) \Rightarrow \mathtt{Goal}_N(b, b, .., b) \tag{10}$$

where $\mathtt{In}_{N,k}(h, h.., h, b, b, .., b)$ becomes

$$\mathtt{Room}(1) \otimes \underbrace{\mathtt{Empty}(h) \otimes \cdot\cdot \otimes \mathtt{Empty}(h)}_{k \text{ times}} \otimes \underbrace{\mathtt{Floor}(1, b) \otimes \cdot\cdot \otimes \mathtt{Floor}(1, b)}_{N \text{ times}}$$

and $\mathtt{Goal}_N(b, b, .., b)$ becomes

$$\underbrace{\mathtt{Floor}(2, b) \otimes \mathtt{Floor}(2, b) \otimes \cdot\cdot \otimes \mathtt{Floor}(2, b)}_{N \text{ times}}$$

The advantage of the trick is that the number of all 'generic configurations' formally generated from the initial 'generic configuration' $\mathtt{In}_{N,k}(h, .., h, b, .., b)$ by means of actions (4)–(6) turns out to be polynomial $O(kN)$. See Fig. 1 where each of the arrows represents a formal application of one of the actions (4)–(6).

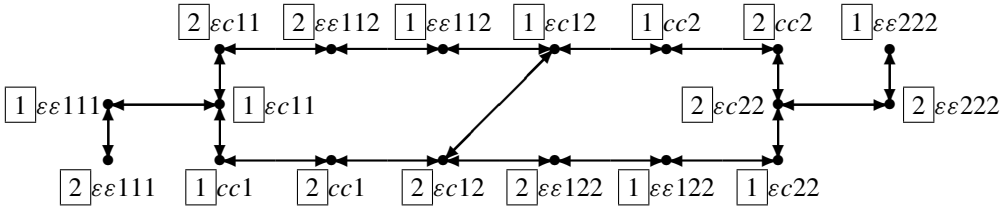


Fig. 1. The 'generic' search space for $N = 3,\ k = 2$. Here "Room(1)" is abbreviated as "1", "Room(2)" as "2", "Empty(h)" as "ε", "Hold(h, b)" as "c", "Floor($1, b$)" as "1", and "Floor($2, b$)" as "2".

The effect is that we can find (in polytime) the *shortest* plan **but** for our 'generic' problem (10) (see Theorem 5.2). Thus, for $N = 3,\ k = 2$, one of the three shortest plans can be taken as:

$$\boxed{1}\varepsilon\varepsilon 111 \to \boxed{1}\varepsilon c11 \to \boxed{1}cc1 \to \boxed{2}cc1 \to \boxed{2}\varepsilon c12 \to$$
$$\boxed{2}\varepsilon\varepsilon 122 \to \boxed{1}\varepsilon\varepsilon 122 \to \boxed{1}\varepsilon c22 \to \boxed{2}\varepsilon c22 \to \boxed{2}\varepsilon\varepsilon 222$$

(1) apply $\mathtt{pick}(1, h, b)$ to the initial 'generic' configuration[4]
$(\mathtt{Room}(1) \otimes \underline{\mathtt{Empty}(h)} \otimes \mathtt{Empty}(h) \otimes \underline{\mathtt{Floor}(1, b)} \otimes \mathtt{Floor}(1, b) \otimes \mathtt{Floor}(1, b))$,
resulting in $(\mathtt{Room}(1) \otimes \mathtt{Hold}(h, b) \otimes \mathtt{Empty}(h) \otimes \mathtt{Floor}(1, b) \otimes \mathtt{Floor}(1, b))$;

(2) apply $\mathtt{pick}(1, h, b)$ to this 'generic' configuration
$(\mathtt{Room}(1) \otimes \mathtt{Hold}(h, b) \otimes \underline{\mathtt{Empty}(h)} \otimes \underline{\mathtt{Floor}(1, b)} \otimes \mathtt{Floor}(1, b))$,
resulting in $(\mathtt{Room}(1) \otimes \mathtt{Hold}(h, b) \otimes \mathtt{Hold}(h, b) \otimes \mathtt{Floor}(1, b))$;

(3) apply $\mathtt{move}(1, 2)$; etc., etc., eventually reaching the 'generic' goal
$(\mathtt{Room}(2) \otimes \mathtt{Empty}(h) \otimes \mathtt{Empty}(h) \otimes \mathtt{Floor}(2, b) \otimes \mathtt{Floor}(2, b) \otimes \mathtt{Floor}(2, b))$.

To perform the second stage within our approach — that is to show how to convert the *mock* plan found into a plan of the actions within the real world, and thereby to *justify* our approach, we individualize all occurrences of the 'generic' names in a coherent way (see Theorems 4.1 and 4.2), to obtain the desired real plan:

(1) apply $\mathtt{pick}(1, h_1, b_1)$ to the real initial configuration
$(\mathtt{Room}(1) \otimes \underline{\mathtt{Empty}(h_1)} \otimes \mathtt{Empty}(h_2) \otimes \underline{\mathtt{Floor}(1, b_1)} \otimes \mathtt{Floor}(1, b_2) \otimes \mathtt{Floor}(1, b_3))$,
resulting in $(\mathtt{Room}(1) \otimes \mathtt{Hold}(h_1, b_1) \otimes \mathtt{Empty}(h_2) \otimes \mathtt{Floor}(1, b_2) \otimes \mathtt{Floor}(1, b_3))$;

(2) apply $\mathtt{pick}(1, h_2, b_2)$ to this real configuration
$(\mathtt{Room}(1) \otimes \mathtt{Hold}(h_1, b_1) \otimes \underline{\mathtt{Empty}(h_2)} \otimes \underline{\mathtt{Floor}(1, b_2)} \otimes \mathtt{Floor}(1, b_3))$,
resulting in $(\mathtt{Room}(1) \otimes \mathtt{Hold}(h_1, b_1) \otimes \mathtt{Hold}(h_2, b_2) \otimes \mathtt{Floor}(1, b_3))$;

(3) apply $\mathtt{move}(1, 2)$; etc., etc., eventually reaching the real goal
$(\mathtt{Room}(2) \otimes \mathtt{Empty}(h_1) \otimes \mathtt{Empty}(h_2) \otimes \mathtt{Floor}(2, b_1) \otimes \mathtt{Floor}(2, b_2) \otimes \mathtt{Floor}(2, b_3))$.

Theorems 4.1 and 4.2 provide an easy-to-check syntactic condition for our *erasing individuality* technique to be *automatically correct* within Example 1.11. (See Corollary 4.1) ■

Example 1.12 illustrates confusing subtleties of *asymmetrical* inputs-outcomes even within the same 'symmetrical' domain of Example 1.11.

Example 1.12. There is a ball in each of two rooms, say ball b_1 in room 1, and ball b_2 in room 2. A one-handed robot is to exchange these two balls.

(a) The 'real' planning problem in Example 1.12 is as follows:

$$(\mathtt{Room}(1) \otimes \mathtt{Empty}(h_1) \otimes \mathtt{Floor}(1, b_1) \otimes \mathtt{Floor}(2, b_2)) \Longrightarrow (\mathtt{Floor}(2, b_1) \otimes \mathtt{Floor}(1, b_2)). \tag{11}$$

(b) Similar to Example 1.11, by *erasing individuality* we abstract it as the 'generic' planning problem:

$$(\mathtt{Room}(1) \otimes \mathtt{Empty}(h) \otimes \mathtt{Floor}(1, b) \otimes \mathtt{Floor}(2, b)) \Longrightarrow (\mathtt{Floor}(2, b) \otimes \mathtt{Floor}(1, b)). \tag{12}$$

[4] The parts involved in the action performance are underlined.

In contrast to Example 1.11, there is a violent discrepancy between solutions to our 'generic' planning problem (12) and solutions to our real planning problem (11):

(a) A solution to this trivial planning problem (12) is evident: "Do nothing".
(b) But such a trivial 'generic' solution cannot give any clue to the real planning problem (11). ■

In Section 4 we intend to remove the mystery of when and why our 'generic' approach can be successfully applied to the planning problems. (See also Example 4.1)

2. Why Linear Logic?

As a logical formalism to specify and sort out planning problems under uncertainty, we use *linear logic* introduced by Girard [16] as a resource-sensitive refinement of traditional logic (see also Section 1.2).

In Table 1 we collect the rules of so-called multiplicative–additive fragment of linear logic, which we are actually dealing with. For a more complete presentation, we refer the reader to [16]. See also [28,22].

2.1. Basic clues from linear logic

We will exploit the following basic ideas from the linear logic paradigm:

(A) Implication as an Action
Within linear logic, a statement of the form

$$X \vdash Y$$

can be interpreted as a specification of a certain action enabled by *precondition* X and resulting in *postcondition* Y, with X being consumed in the process; and vice versa. In other words, its application to a current 'configuration' presupposes both negative and positive effects: first, 'deleting' X from the current 'configuration', and then getting the next 'configuration' by 'adding' Y.

(B) Sets versus Multisets.
The traditional first-order theories deal with *sets*, and, therefore, they are based on the traditional Boolean interpretation for formulas built up from atomic formulas like $Q(z)$ or $(z \in q)$, with the Boolean connectives $\wedge$ and $\vee$ being polysemantic in some respects.

As opposed to traditional logics, linear logic is capable of directly handling *multisets* by refining the traditional $\wedge$ connective into two connectives $\otimes$ (*tensor*) and & (*with*):

(i) The intended meaning of $A \otimes B$ is *"A and B co-exist together"*.
(ii) Whereas $A \& B$ is intended to represent *"any of A and B"*.

For instance, the fact that *"two copies of b have property Q"* can be directly expressed as the formula $(Q(b) \otimes Q(b))$ within the linear logic framework.

Comment 2.1. To be fair, linear logic interpretations are much more subtle.

E.g. a formula of the form $(P(b) \otimes Q(b))$ can be interpreted (*and used!*) in two different ways:

(1) *"one and the same copy of b has both properties P and Q"*, or
(2) *"one copy of b has property P, and another has property Q"*.

Moreover,

(1) Specifying *actions*, we intend to use the first interpretation.
Notice that we have to take precautions against making an error.
For instance, in the case of a system with a single action of the form

$$(P(z) \otimes Q(z)) \vdash R(z), \tag{13}$$

the above *erasing individuality* trick generally fails.
In particular, a 'real' planning problem of the form

$$(P(b_1) \otimes Q(b_2)) \Rightarrow (R(b_1) \oplus R(b_2))$$

has no solution, notwithstanding that the corresponding problem of the 'generic' form

$$(P(b) \otimes Q(b)) \Rightarrow (R(b) \oplus R(b))$$

has a one-step solution, namely, "apply action (13)".

Table 1
The inference rules of intuitionistic affine logic

I	$\dfrac{}{A \vdash A}$	**P**	$\dfrac{\Gamma,\ A, B,\ \Delta \vdash C}{\Gamma,\ B, A,\ \Delta \vdash C}$
Cut	$\dfrac{\Gamma \vdash A \quad A, \Delta \vdash C}{\Gamma, \Delta \vdash C}$		
L⊗	$\dfrac{\Gamma, A, B \vdash C}{\Gamma, (A \otimes B) \vdash C}$	**R⊗**	$\dfrac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash (A \otimes B)}$
L&	$\dfrac{\Gamma, A \vdash C}{\Gamma, (A\&B) \vdash C} \quad \dfrac{\Gamma, B \vdash C}{\Gamma, (A\&B) \vdash C}$	**R&**	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash (A\&B)}$
L⊕	$\dfrac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, (A \oplus B) \vdash C}$	**R⊕**	$\dfrac{\Gamma \vdash A}{\Gamma \vdash (A \oplus B)} \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash (A \oplus B)}$
L⊸	$\dfrac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, (A \multimap B), \Delta \vdash C}$	**R⊸**	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash (A \multimap B)}$
L1	$\dfrac{\Gamma \vdash C}{\Gamma, 1 \vdash C}$	**R1**	$\dfrac{}{\vdash 1}$
W	$\dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}$		

(2) The second interpretation is intended to be used in the case where $(P(b) \otimes Q(b))$ describes a 'generic' *configuration* of the system.

To circumvent the conflict of these mutually exclusive tendencies, we will invoke actions that are *monadic* with respect to such a z. (See Definition 4.4) ■

(C) A branching behaviour of $\oplus$, the dual to &.

Dually to &, $A \oplus B$ is intended to represent *"either A or B"*.

This allows us to handle *uncertainty* about the effects of actions in a direct way.

Thus the performance of an action represented by the following *disjunctive* form

$$X \vdash (A \oplus B)$$

yields a *non-deterministic* effect:

- X is transformed either into A or into B.

From a point of view of the game 'Robot against Nature',

(a) a robot's move is to choose one of the actions, say $X \vdash (A \oplus B)$, to be performed;

(b) an opponent's move is to respond with A, or to respond with B.

2.2. Disjunctive Horn linear logic

We will handle the planning aspects of AI systems in purely logical terms, where the difficulties of being over-complicated are obviated within the framework of a *Horn fragment* of linear logic in the sense of [19].

At the very beginning, we use Horn linear logic as a *specification language* for describing the robot systems in question. The actions performed by a robot are specified in terms of Horn axioms introduced below in Definition 2.2: *disjunctive* Horn axioms are used for specifying the actions with *non-deterministic effects*.

The *planning* problem is represented as a sequent of the form $W \vdash \widetilde{Z}$, where W specifies a given initial situation, and $\widetilde{Z}$ describes the desired set of final partial situations [28].

We will use the following syntactic conventions.

Definition 2.1. A number of *sorts* $\tau_1,..,\tau_s$ are assumed.
A formula of the form

$$Q_1(t_{1,1}, .., t_{1,k_1}) \otimes \cdots \otimes Q_m(t_{m,1}, .., t_{m,k_m})$$

where $Q_1,..,Q_m$ are predicate symbols, $t_{1,1},..,t_{m,k_m}$ are variables or constants of some sorts, is called an *LL-monomial* of *degree* m. The *trivial* LL-monomial V of *degree* 0 is allowed, for which $X \otimes V = X$.

Formulas of the form $(Z_1 \oplus \cdots \oplus Z_k)$, with $Z_1,..,Z_k$ being LL-monomials, are called *LL-polynomials*.

LL-monomials and LL-polynomials are taken modulo commutativity and associativity of $\otimes$ and $\oplus$.

Definition 2.2. A *disjunctive Horn sequent* is a sequent of the form

$$X \vdash (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m)$$

where X, Y_1,Y_2,..,Y_m are LL-monomials $(m \geq 1)$.

Intuitively, such a Horn sequent represents a robot's *move* enabled with *precondition* X accomplished by one of the possible *responses* $Y_1, Y_2, \ldots, Y_m$ of Nature.

A Horn sequent of the form $X \vdash Y_1$ will be also called a *pure Horn sequent*.

2.3. HLL theories and HLL derivation rules

Definition 2.3. A *theory* T is specified by means of a recursive set of its 'axiom sequents' (denoted by $\mathtt{Ax}_T$). We will call these sequents 'proper axioms' of T to distinguish them from standard logical axioms like $A \vdash A$.

A *proof within a theory* T is a finite list of sequents (without repetitions) in which each sequent is the result of applying a derivation rule from a given logical system to sequents appearing earlier in the list.

If the proof terminates with the sequent $W \vdash Z$ then $W \vdash Z$ is said to be proved within theory T.

Definition 2.4. For a theory T such that its set of 'proper axioms' $\mathtt{Ax}_T$ consists of disjunctive Horn sequents, we introduce the following derivation rules:

(a) **Horn rule**:
For $X \vdash Y$, an instance of a sequent from $\mathtt{Ax}_T$, and any LL-monomial V,

$$\frac{(V \otimes Y) \vdash \widetilde{Z}}{(V \otimes X) \vdash \widetilde{Z}} \tag{14}$$

(b) $\oplus$**-Horn rule**:
For $X \vdash (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m)$, an instance of a sequent from $\mathtt{Ax}_T$, and any LL-monomial V,

$$\frac{(V \otimes Y_1) \vdash \widetilde{Z} \qquad (V \otimes Y_2) \vdash \widetilde{Z} \quad \ldots \quad (V \otimes Y_m) \vdash \widetilde{Z}}{(V \otimes X) \vdash \widetilde{Z}} \tag{15}$$

(c) **Logical Axioms**:
For any LL-monomial Z' and $1 \leq i \leq k$,

$$\frac{}{(Z_i \otimes Z') \vdash \widetilde{Z}} \tag{16}$$

Here $\widetilde{Z}$ is an LL-polynomial of the form $(Z_1 \oplus \cdots \oplus Z_i \oplus \cdots \oplus Z_k)$.

The rules are taken modulo commutativity and associativity of $\otimes$ and $\oplus$.

Comment 2.2.

(a) A rule of the form (14) can be conceived as an abbreviated version of the linear logic rule

$$\frac{(V \otimes Y) \vdash \widetilde{Z} \qquad X \vdash Y}{(V \otimes X) \vdash \widetilde{Z}}$$

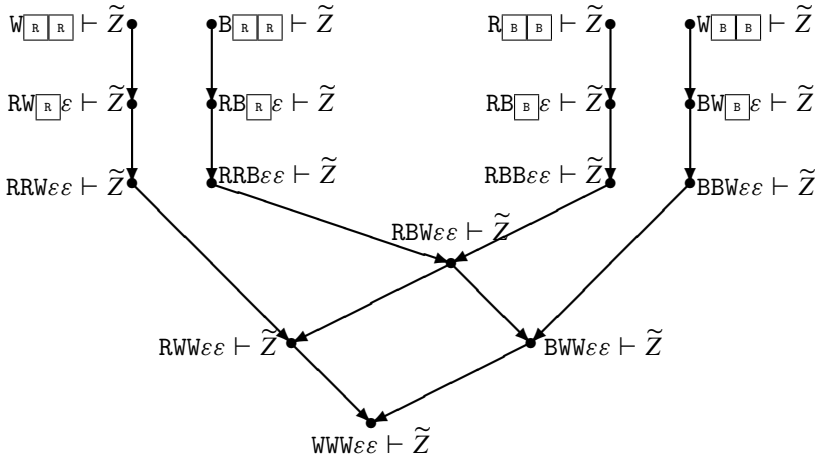


Fig. 2. An HLL-canonical derivation for $\mathtt{WWW}\varepsilon\varepsilon \vdash \widetilde{Z}$ in Example 4.2, with $N=3$, $k=2$. By $\widetilde{Z}$ we denote our goal $\boxed{\mathtt{B}}\boxed{\mathtt{B}} \oplus \boxed{\mathtt{R}}\boxed{\mathtt{R}}$.

(b) A rule of the form (15) is an abbreviated version of the linear logic rule

$$\frac{(V \otimes Y_1) \vdash \widetilde{Z} \qquad (V \otimes Y_2) \vdash \widetilde{Z} \quad \ldots \quad (V \otimes Y_m) \vdash \widetilde{Z} \qquad X \vdash (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m)}{(V \otimes X) \vdash \widetilde{Z}}$$

(c) A rule of the form (16) is a weak version of the Weakening Rule applied only to standard axioms as $X \vdash X$, and, actually, it belongs to affine logic.

The *relative completeness* of the above HLL rules is stated in the following strong version of the *normalization lemma* for Horn linear logic (cf. [20]).

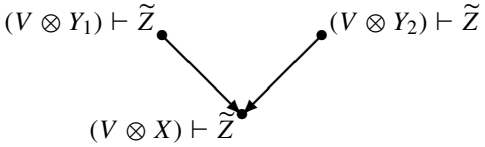
Lemma 2.1. *Let T be a theory such that the set of its 'proper axioms' $\mathtt{Ax}_T$ consists of disjunctive Horn sequents.*

Given an LL-monomial W and an LL-polynomial $\widetilde{Z}$, a sequent of the form $W \vdash \widetilde{Z}$ is derivable from $\mathtt{Ax}_T$ by the inference rules of affine logic if and only if the sequent $W \vdash \widetilde{Z}$ can be proved within theory T equipped only with the above derivation rules (14), (15)*, and* (16)*.*

For the sake of clarity and non-redundancy in our examples like Fig. 2, a rule of the form (14) or (15), say

$$\frac{(V \otimes Y_1) \vdash \widetilde{Z} \qquad (V \otimes Y_2) \vdash \widetilde{Z}}{(V \otimes X) \vdash \widetilde{Z}}$$

for $X \vdash (Y_1 \oplus Y_2)$ being an instance of a proper axiom of theory T, will be depicted as:



This gives rise to the following definition of 'folded' derivations within theory T, with identical parts being glued together (see also Fig. 2):

Definition 2.5. Given a theory T such that the set of its 'proper axioms' $\mathtt{Ax}_T$ consists of disjunctive Horn sequents, an *HLL-canonical* representation of a proof within T is defined as a finite directed graph having no directed cycles such that

(a) each node v with no incoming edges is labelled by a logical axiom of the form (16);
(b) each node v with incoming edges (w_1, v), .., (w_m, v) is labelled by a sequent, say $\Gamma \vdash \Delta$, so that $\Gamma \vdash \Delta$ is obtained by applying a rule of the form (14) or (15) to the sequents $\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_m \vdash \Delta_m$ that label the above nodes w_1, .., w_m.

2.4. The extreme expressive power of Horn linear logic

At first glance, the class of planning problems expressible in Horn linear logic seems to be a very specific subclass of those planning problems that can be expressed, say, in STRIPS with its 'deletion–addition' effects (see Section 1.3), since, for instance, neither negations nor embedded implications appear in the HLL representation of problems.

Recall that any STRIPS action, say, assuming the *precondition* A_1, A_2, and A_3, *delete* A_1, and then *add* B_1, is simulated by a linear logic sequent of the Horn-like form

$$(A_1 \otimes A_2 \otimes A_3) \vdash (B_1 \otimes A_2 \otimes A_3).$$

It turns out that the linear logic formalism perfectly captures the 'deletion–addition' effects of STRIPS: the reason is that within linear logic, whatever linear logic sequent

$$\Gamma \vdash \Delta$$

we take, its application to a current 'configuration' always presupposes both negative and positive effects: first, 'deleting' Γ from the current 'configuration', and then obtaining the next 'configuration' by 'adding' Δ.

Furthermore, from the complexity point of view, Horn linear logic is much more expressive than we need for the purposes of STRIPS simulation:

(a) pure Horn linear logic simulates the reachability problem for Petri nets [15];
(b) disjunctive Horn linear logic even simulates Minsky machines [21] and, hence, is undecidable;
(c) linear logic with weakening (a.k.a. affine logic) is decidable [25,26], but the Horn fragments of affine logic are still very complex.

To obviate the problem of overestimating complexity bounds, in [22] we have shown that some well-known AI planning problems, in particular, the whole of STRIPS, can be faithfully represented within Horn linear logic syntactically restricted to well-balanced axioms (in which the 'size' of the left-hand side equals the 'size' of the right-hand side).

Along the way, Horn linear logic machinery guarantees PSPACE and EXPTIME upper bounds for planning in deterministic and non-deterministic domains, respectively (cf. Section 5).

The formal syntactic similarity — that the Horn fragment of Boolean logic and the pure Horn fragment of propositional linear logic use the same formulas, might have caused a certain misunderstanding about their comparative expressive power for AI purposes. To clear up this point,

(a) the expressive power of the Horn fragment of Boolean logic is very weak (it is good, for instance, only for propositional STRIPS without 'deletion' effects);
(b) whereas the Horn fragment of propositional linear logic is extremely powerful because of the refined linear logic machinery (on top of that, we impose additional syntactic constraints on HLL in order to *lower* its power to the complexity level of the whole of STRIPS).

3. Planning versus decidability: Plans $\Longleftrightarrow$ LL proofs

It should be pointed out that we are looking for plans; hence, a simple decision procedure is not a satisfactory solution (the simple existence of a plan is almost evident in many practical cases). Compared to many existing logic formalisms for planning, the advantage of linear logic here is that it provides a direct and clear correspondence between proofs and plans [28,29,19,22].

3.1. Plans as programs. Strong and weak plans

According to Definition 1.2, for systems with pure deterministic actions, a plan $\mathcal{P}$ is defined as a chain of *commands*, or *moves*, to perform certain actions [30,3,28], something like:

(i) apply α_{i_1}; go to the next command;
(ii) apply α_{i_2}; go to the next command;

(iii) .....;
(iv) apply α_{i_n}; go to the next command;
(v) halt;

In order to cover the general case where actions with *non-deterministic effects* are allowed, we extend this definition to plans $\mathcal{P}$, in which a *command* prescribing the execution of such an action should anticipate all possible responses of Nature.

Definition 3.1. Suppose that the actions within a given system are specified with disjunctive Horn sequents from $\mathtt{Ax}_T$, the set of 'proper axioms' of the theory T.

A *plan* $\mathcal{P}$ based on $\mathtt{Ax}_T$ is a set of *labeled commands*, with one command being designated as the *initial* one. The following *labelled commands* are invoked here:

(a) A *command* is of the form:

$$\boxed{\begin{array}{l} l\text{: Given a state of the form } X \otimes V, \\ \quad \text{apply the action specified with } X \vdash (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m), \\ \qquad \text{which is an instance of a sequent from } \mathtt{Ax}_T; \\ \quad \text{upon getting response } Y_1, \text{ go to the command labelled with } l', \\ \quad \text{upon getting response } Y_2, \text{ go to } l'', \\ \quad \ldots\ldots \\ \quad \text{upon getting response } Y_m, \text{ go to } l^{(m)}. \end{array}} \tag{17}$$

(b) A *command with parameters*, or a *command schema*, is of the following form:

$$\boxed{\begin{array}{l} l\text{: Let } D_1,..,D_\ell \text{ be sets of constants of the theory } T; \\ \quad \text{given a state of the form } X(d_1, ..., d_\ell) \otimes V \\ \qquad \text{where } d_1 \in D_1, \ldots, d_\ell \in D_\ell, \\ \quad \text{apply the action specified with} \\ \qquad X(d_1, ..., d_\ell) \vdash (Y_1(d_1, ..., d_\ell) \oplus Y_2(d_1, ..., d_\ell) \oplus \cdots \oplus Y_m(d_1, ..., d_\ell)), \\ \qquad \text{which is an instance of a sequent from } \mathtt{Ax}_T; \\ \quad \text{upon getting response } Y_1(d_1, ..., d_\ell), \text{ go to the command labeled with } l', \\ \quad \text{upon getting response } Y_2(d_1, ..., d_\ell), \text{ go to } l'', \\ \quad \ldots\ldots \\ \quad \text{upon getting response } Y_m(d_1, ..., d_\ell), \text{ go to } l^{(m)}. \end{array}} \tag{18}$$

The *command schema* (18) is a generalized version of (17).

(c) A *halting command* is of the form

$$\boxed{l_0 : \mathit{halt}}. \tag{19}$$

The commands in plan $\mathcal{P}$ are assumed to have pairwise distinct labels.

Taking labels as nodes and making arrows from a label l to each of the labels $l', l'', \ldots, l^{(m)}$ in accordance with the corresponding command of the form (17) or (18), we obtain the *control chart* of plan $\mathcal{P}$.

An additional assumption is that the control chart of $\mathcal{P}$ must form an acyclic graph: no loop is allowed in the control chart of $\mathcal{P}$. ■

Notice that the case of actions with deterministic effects is covered by (18) by letting $m = 1$:

$$\boxed{\begin{array}{l} l\text{: Let } D_1,..,D_\ell \text{ be sets of constants of the theory } T; \\ \quad \text{given a state of the form } X(d_1, ..., d_\ell) \otimes V \\ \qquad \text{where } d_1 \in D_1, \ldots, d_\ell \in D_\ell, \\ \quad \text{apply the action specified with } X(d_1, ..., d_\ell) \vdash Y(d_1, ..., d_\ell), \\ \qquad \text{which is an instance of a sequent from } \mathtt{Ax}_T; \\ \quad \text{go to the command labelled } l'. \end{array}} \tag{20}$$

Comment 3.1. The classical definition of branching plans (see, for instance, [4]) assumes their tree-like structure, which implies generally their exponential size. The concise representation of plans above as *'acyclic programs with parameterized commands'* allows us to show the difference between situations in which the winning strategy in the form of a game tree is exponential *per se,* and situations in which the winning strategy in the form of a game tree is exponential only because the number of different positions that should be mentioned within the game tree happens to be exponential. For instance, for any game on the $N \times N$-board, its game trees are generally exponential regardless to the complexity of the game itself. (Cf. Example 4.2)

Invoking commands *with parameters* is usual practice in many cases:

(a) Like calculi where one axiom with free variables represents a potentially infinite number of instances obtained by replacing free occurrences of variables in the axiom with any constants, a *command schema* of the form (18) is introduced to represent a set of 'command instances' dealing with constants ranging over the prescribed domains D_1,..,D_ℓ.

(b) It is quite typical to game strategies. For instance, in chess a move to promote a pawn to a queen can be recorded in the algebraic notation as "$x7$-$x8$Q" where $x \in \{a, b, c, d, e, f, g, h\}$.

(c) Common-sense instructions such as "take *any* ball" (meaning no matter which one from a given set), are typical within solutions made by humans.

An unrestricted choice of values for parameters within a command schema seems to cause a non-deterministic behaviour of our plans. But we will require that a *correct* plan must yield the correct result under whichever choice of admissible values of the parameters is made.

Definition 3.2. Given an initial configuration specified with an LL-monomial W, we can develop a *game tree* according to a plan $\mathcal{P}$ as follows.

First, we define a *partial game tree* as a rooted tree, in which each of its nodes is labelled by a pair (U, l) where U is a configuration of the system, and l labels a command in $\mathcal{P}$, so that

(a) The root is labeled by (W, l_{init}) where l_{init} is the label of the initial command in $\mathcal{P}$.

(b) Only terminal nodes may be labelled by (U, l_0) where l_0 is the label of a halting command in $\mathcal{P}$.

(c) For each non-terminal node v, labeled by (U, l) where l is the label of a command of the form (18), the following holds:

(c1) There are $d_1 \in D_1$,.., $d_\ell \in D_\ell$ such that U is of the form $X(d_1, ..., d_\ell) \otimes V$;

(c2) v must have exactly m children, say w_1,.., w_m, labelled by (U_1, l'),..., $(U_m, l^{(m)})$, resp., such that each U_j is of the form $Y_j(d_1, ..., d_\ell) \otimes V$, for $j = 1, .., m$.

(Recall that the formal performance of the action in question is to replace precondition X with some postcondition Y_j, and thereby the U_j's represent all possible effects of the action).

We will say that the partial game tree *ends in* $\{Z_1, .., Z_k\}$ if each of its branches ends in a node labelled by (U, l_0) such that l_0 is the label of a halting command in $\mathcal{P}$, and U is of the form $(Z_i \otimes Z')$, for some $i = 1, .., k$.

Definition 3.3. Let W specify an initial situation and Z_1,.., Z_k represent a set of goal situations.

A plan $\mathcal{P}$ based on $\mathtt{Ax}_T$ is called a *strong solution* to a planning problem

$$W \Rightarrow (Z_1 \oplus \cdots \oplus Z_k),$$

if any partial game tree developed according to plan $\mathcal{P}$ for the given initial W, can be completed to a game tree $\mathcal{T}$ so that $\mathcal{T}$ ends in $\{Z_1, .., Z_k\}$.

Definition 3.3 guarantees a strong plan will achieve the goal under any circumstances and without *deadlocks*: the plan envisages *all* possible reactions of Nature on the road from the initial situation W to any final one from the set Z_1,.., Z_k.

Comment 3.2. For the case where each action has a deterministic effect, Definition 3.3 provides the so-called *disjunctive property* (see Corollary 7.1).

The weakest form of a solution — that is achieving the goal under extremely favorable circumstances, is given below:

Definition 3.4. Let W specify an initial situation and Z_1,.., Z_k represent a set of goal situations.

We will say that a plan $\mathcal{P}$ based on $\mathtt{Ax}_T$ is a *weak solution* to a planning problem

$$W \Rightarrow (Z_1 \oplus \cdots \oplus Z_k),$$

if there is a partial game tree developed according to plan $\mathcal{P}$ for the given initial W, such that at least one of its branches terminates in a node labelled by (U, l_0) with l_0, the label of a halting command in $\mathcal{P}$, and U of the form $(Z_i \otimes Z')$, for some $i = 1, .., k$.

In Section 6 we will resolve the weak planning problem as well.

As for the strong planning problem, we rely upon the following direct correlation between complete game trees and derivations in Horn linear logic.

Theorem 3.1 (*cf. Kanovich [19]*). *Let T be a theory such that the set of its 'proper axioms' $\mathtt{Ax}_T$ consists of disjunctive Horn sequents.*

Given an LL-monomial W and an LL-polynomial $(Z_1 \oplus \cdots \oplus Z_k)$, the following three statements are equivalent:

(a) *There exists a complete game tree based on actions specified in $\mathtt{Ax}_T$ leading from W to the set of goals $\{Z_1, .., Z_k\}$;*
(b) *A sequent of the form $W \vdash (Z_1 \oplus \cdots \oplus Z_k)$ is derivable from $\mathtt{Ax}_T$ by the inference rules of affine logic;*
(c) *A sequent of the form $W \vdash (Z_1 \oplus \cdots \oplus Z_k)$ can be proved within theory T equipped only with the Horn linear logic rules (14), (15), and (16). (See Lemma 2.1)*

4. 'Generic' proofs $\Longrightarrow$ 'real' plans

Definition 4.1. Let A be a formula that may contain variables. For a fixed sort τ, we will write $A(z_1, z_2, .., z_n)$ to indicate distinguished variables z_1, z_2,.., z_n of sort τ that may occur in the formula A. The *symmetrical closure* $A^{Sym}(z_1, z_2, .., z_n)$ of $A(z_1, z_2, .., z_n)$ is defined as:

$$A^{Sym}(z_1, z_2, .., z_n) := \bigoplus_{\pi} A(z_{\pi(1)}, z_{\pi(2)}, .., z_{\pi(n)})$$

where π ranges over all permutations of $\{1, .., n\}$.

A formula $A(z_1, z_2, .., z_n)$ is *symmetrical* if it is equivalent to its symmetrical closure, that is, for any permutation π, a sequent of the form

$$A(z_1, z_2, .., z_n) \vdash A(z_{\pi(1)}, z_{\pi(2)}, .., z_{\pi(n)})$$

is derivable in linear logic.

Definition 4.2. Let $\#_\tau(X)$ denote the number of occurrences of variables of sort τ in an LL-monomial X.
For an LL-polynomial $Y_1 \oplus \cdots \oplus Y_m$,

$$\#_\tau(Y_1 \oplus \cdots \oplus Y_m) := \max\{\#_\tau(Y_j) \mid j = 1, 2, .., m\}.$$

The desired correlation "generic proofs $\Longleftrightarrow$ real proofs" is explained in Definition 4.3.

Definition 4.3. Let $\mathtt{Ax}_T$ be a set of 'proper axioms' specifying the theory T.

A given *sort* τ will be called **generic** within theory T if for any variables z, z_1, z_2,.., z_n of sort τ, an LL-monomial $W(z_1, z_2, .., z_n)$, in which each of the z_i has exactly one occurrence, and an LL-polynomial $\widetilde{Z}(z_1, z_2, .., z_n)$ such that $\#_{z_i}(\widetilde{Z}) \leq 1$ for each z_i, the following holds:
If a sequent of the form

$$W(z, z, .., z) \vdash \widetilde{Z}(z, z, .., z) \tag{21}$$

is derivable from $\mathtt{Ax}_T$ by the rules of affine logic, then a sequent of the form

$$W(z_1, z_2, .., z_n) \vdash \widetilde{Z}^{Sym}(z_1, z_2, .., z_n) \tag{22}$$

is also derivable from $\mathtt{Ax}_T$ by the rules of affine logic.

Comment 4.1.

(a) The condition on the number of occurrences of z_i makes sense.
Let, for instance,

$$W(z_1, z_2) := (P(z_1) \otimes Q(z_2)),$$

and

$$Z(z_1, z_2) := (P(z_1) \otimes Q(z_1)).$$

Then

$$W(z, z) \vdash Z(z, z)$$

is derivable, but

$$W(z_1, z_2) \vdash (Z(z_1, z_2) \oplus Z(z_2, z_1))$$

is not.

(b) We have need of the permutations and the symmetrical closure ^{Sym} in Definition 4.3.
Let, for instance,

$$W(z_1, z_2) := (P(z_1) \otimes Q(z_2)),$$

and

$$Z(z_1, z_2) := (Q(z_1) \otimes P(z_2)).$$

Then

$$W(z, z) \vdash Z(z, z)$$

is derivable, but

$$W(z_1, z_2) \vdash Z(z_1, z_2)$$

is not. ■

By replacing variables with constants in Definition 4.3, we obtain the following proposition:

Proposition 4.1. *Let τ be a generic sort within a theory T.*

Let $z_1,z_2,..,z_n$ be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial such that $\#_{z_i}(\widetilde{Z}) \leq 1$ for each z_i.

Then for any constants $b, b_1,b_2,..,b_n$ of sort τ that have no occurrence in $\mathtt{Ax}_T$, W, and $\widetilde{Z}$, the sequent

$$W(b_1, b_2, .., b_n) \vdash \widetilde{Z}^{Sym}(b_1, b_2, .., b_n) \tag{23}$$

is derivable from $\mathtt{Ax}_T$ in affine logic if and only if a sequent of the 'generic' form

$$W(b, b, .., b) \vdash \widetilde{Z}(b, b, .., b) \tag{24}$$

is derivable from $\mathtt{Ax}_T$ by the rules of affine logic.

Informally, Proposition 4.1 says that in the case of a generic sort τ, the *original* planning problem (23) dealing with a variety of n 'real objects' can be fully sorted out in terms of the 'generic' planning problem (24) dealing with only *one* 'generic object'.

To guarantee the main hypothesis of Proposition 4.1 — that τ is a generic sort within a theory T, we rely upon the following syntactic property of the proper axioms of T.

Definition 4.4. Given a sort τ, a disjunctive Horn sequent $X \vdash (Y_1 \oplus \cdots \oplus Y_m)$ is said to be τ-*monadic* if

$$\#_\tau(Y_1 \oplus \cdots \oplus Y_m) \leq \#_\tau(X) \leq 1.$$

The following theorem plays the key role on the road from 'generic proofs' to 'real proofs'.

Theorem 4.1. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents. Then the sort τ is generic within the theory T.*

The following lemma gives the important information about derivations:

Lemma 4.1. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents.*

Let $z_1, z_2,.., z_n$ be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial.
Suppose that D is a derivation in HLL-theory T developed for the sequent

$$W(z, z, .., z) \vdash \widetilde{Z}(z, z, .., z). \tag{25}$$

Then every sequent occurring in D is of the form $U(z, .., z) \vdash \widetilde{Z}(z, .., z)$, for some LL-monomial $U(z_1, z_2, .., z_n)$ in which each of the z_i has at most one occurrence.

Proof. This follows by passing through D bottom-up: from the root sequent (25) to the leaves, which are logical axioms (16). Take into account that our rules (14) and (15) preserve the inductive statement in the case of τ-monadic axioms from $\mathtt{Ax}_T$. ■

Proof of Theorem 4.1. Given an affine logic proof for (21), first, by Lemma 2.1 we translate it into a derivation with rules (14), (15), and (16).

Then we assemble the desired proof for (22) by the top-down induction.

It suffices to examine the most complicated case where rule (14) invokes an instance of a τ-monadic Horn sequent from $\mathtt{Ax}_T$, say $X(v) \vdash (Y_1(v) \oplus Y_2(v))$, with v a variable of sort τ, and produces a sequent of the form

$$V(z, .., z) \otimes X(z) \vdash \widetilde{Z}(z, z, .., z)$$

from sequents of the form

$$V(z, .., z) \otimes Y_1(z) \vdash \widetilde{Z}(z, z, .., z)$$

and

$$V(z, .., z) \otimes Y_2(z) \vdash \widetilde{Z}(z, z, .., z).$$

Lemma 4.1 implies that $V(z, .., z)$ is an instance of an LL-monomial $V(z_2, .., z_n)$ in which each of the $z_2,.., z_n$ has at most one occurrence.

By the inductive hypothesis, both sequents of the form

$$V(u_2, .., u_n) \otimes Y_1(u_1) \vdash \widetilde{Z}^{Sym}(u_1, u_2, .., u_n)$$

and

$$V(v_2, .., v_n) \otimes Y_2(v_1) \vdash \widetilde{Z}^{Sym}(v_1, v_2, .., v_n)$$

are derivable by HLL-rules, where $u_1,..,u_n$, and $v_1,..,v_n$ are permutations of $z_1,..,z_n$.
Since $\widetilde{Z}^{Sym}(z_1, z_2, .., z_n)$ is symmetrical,

$$V(z_2, .., z_n) \otimes Y_1(z_1) \vdash \widetilde{Z}^{Sym}(z_1, z_2, .., z_n)$$

and

$$V(z_2, .., z_n) \otimes Y_2(z_1) \vdash \widetilde{Z}^{Sym}(z_1, z_2, .., z_n)$$

are also HLL-derivable, and the same rule (14) yields the derivability of

$$V(z_2, .., z_n) \otimes X(z_1) \vdash \widetilde{Z}^{Sym}(z_1, z_2, .., z_n).$$

All other cases are proved in a similar way. ■

With Theorem 4.2 we obtain a basic technical tool to convert any 'generic proof' into a 'real plan'.

Theorem 4.2. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents.*

Let $z_1, z_2, .., z_n$ be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial such that for each z_i, $\#_{z_i}(\widetilde{Z}) \leq 1$.

Let b, b_1, b_2,.., b_n be constants of sort τ having no occurrence in $\mathtt{Ax}_T$, W, and $\widetilde{Z}$.

Then every cut-free affine logic proof within theory T for a sequent of the form

$$W(b, b, .., b) \vdash \widetilde{Z}(b, b, .., b), \tag{26}$$

which deals with one *'generic object' b, can be converted (in* `polytime`*) into a strong solution to the original planning problem specified as*

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}^{Sym}(b_1, b_2, .., b_n), \tag{27}$$

which deals with n 'real objects'.

Proof. By Lemma 2.1, a given affine logic proof for sequent (26) can be transformed into an HLL-canonical derivation D having no repetitions. According to Lemma 4.1, each of the nodes in this D is of the form

$$U(b, .., b) \vdash \widetilde{Z}(b, .., b),$$

for some LL-monomial $U(z_1, .., z_n)$.

The strong plan $\mathcal{P}$ we are looking for is assembled as a list of commands introduced step-by-step in the following top-down manner.

With each of the nodes $U(b, .., b) \vdash \widetilde{Z}(b, .., b)$ occurring in D we will associate the following command labelled by l_U:

For a logical axiom $U(b, .., b) \vdash \widetilde{Z}(b, .., b)$, the associated command is defined to be

$$\boxed{l_U : halt}.$$

Suppose that $U(b, .., b) \vdash \widetilde{Z}(b, .., b)$ has been produced by rule (14) from preceding sequents

$$U_1(b, .., b) \vdash \widetilde{Z}(b, .., b), \quad \ldots \quad U_m(b, .., b) \vdash \widetilde{Z}(b, .., b),$$

invoking an instance of a τ-monadic sequent from $\mathtt{Ax}_T$ of the form

$$X(v) \vdash (Y_1(v) \oplus \cdots \oplus Y_m(v)) \tag{28}$$

Lemma 4.1 yields that, for some LL-monomial $V(z_2, .., z_n)$, this $U(b, .., b)$ is of the form $V(b, .., b) \otimes X(b)$, and each of the $U_j(b, .., b)$ is of the form $V(b, .., b) \otimes Y_j(b)$, respectively.
We associate the following command with this node $U(b, .., b) \vdash \widetilde{Z}(b, .., b)$:

$$\boxed{\begin{array}{l} l_U\text{: Given a state of the form } X(d) \otimes V' \\ \qquad \text{where } d \in \{b_1, b_2, .., b_n\}, \\ \text{apply the action specified with } X(d) \vdash (Y_1(d) \oplus \cdots \oplus Y_m(d)); \\ \text{upon getting response } Y_1(d), \text{ go to the command labelled with } l_{U_1}, \\ \ldots\ldots \\ \text{upon getting response } Y_m(d), \text{ go to } l_{U_m}. \end{array}} \tag{29}$$

The command associated with the root $W(b, .., b) \vdash \widetilde{Z}(b, .., b)$ is designated to be the initial one in our plan $\mathcal{P}$.

It remains to prove that the plan $\mathcal{P}$ just assembled is a strong solution to the original planning problem

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}^{Sym}(b_1, b_2, .., b_n).$$

In order to develop an appropriate induction, we assign an 'intermediate' plan $\mathcal{P}_U$ to each node of the form

$$U(b, .., b) \vdash \widetilde{Z}(b, .., b).$$

Namely, $\mathcal{P}_U$ consists of all commands associated with ancestors of this node; the command labeled with l_U is designated to be the initial command of $\mathcal{P}_U$. (Notice that any command of the form (29) refers only to commands associated to *ancestors* of the node in question.)

By induction on D we prove that every plan $\mathcal{P}_U$ is a strong solution to the 'intermediate' planning problem

$$U(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}^{Sym}(b_1, b_2, .., b_n) \tag{30}$$

Given $U(b_1, .., b_n)$ as an initial configuration, let T be an arbitrary partial game tree developed according to plan $\mathcal{P}_U$.

The root of T, say v_0, is labelled by $(U(b_1, .., b_n), l_U)$, which means that the command labelled with l_U must be performed at the first execution step.

It suffices to examine only a non-trivial case where our command is of the form (29), which, in particular, presupposes that $U(b, .., b) \vdash \widetilde{Z}(b, .., b)$ has been produced by rule (14) from the sequents just above

$$U_1(b, .., b) \vdash \widetilde{Z}(b, .., b), \quad \ldots \quad U_m(b, .., b) \vdash \widetilde{Z}(b, .., b),$$

invoking an instance of a sequent from $\mathtt{Ax}_T$ of the form

$$X(v) \vdash (Y_1(v) \oplus \cdots \oplus Y_m(v)) \tag{31}$$

and, for some LL-monomial $V(z_2, .., z_n)$ (see Lemma 4.1),

(a) $U(b, .., b)$ is of the form $X(b) \otimes V(b, .., b)$, and
(b) each of the $U_j(b, .., b)$ is of the form $Y_j(b) \otimes V(b, .., b)$.

Since the command labelled with l_U has been executed with some d chosen among $b_1, .., b_n$, the root v_0 of the game tree T must have exactly m sons, say w_1,.., w_m, such that, for some permutation π:

(a) Root v_0 is labelled by $((X(b_{\pi(1)}) \otimes V(b_{\pi(2)}, .., b_{\pi(n)})),\ l_U)$;
(b) Each of its sons w_j is labelled by $((Y_j(b_{\pi(1)}) \otimes V(b_{\pi(2)}, .., b_{\pi(n)})),\ l_{U_j})$, respectively.

Let $T_1, .., T_m$ be subtrees of T with the roots w_1,.., w_m, respectively. It is readily seen that each T_j can be viewed as a partial game tree developed according to plan $\mathcal{P}_{U_j}$ for an initial configuration of the form

$$Y_j(b_{\pi(1)}) \otimes V(b_{\pi(2)}, .., b_{\pi(n)}).$$

By the inductive hypothesis, each plan $\mathcal{P}_{U_j}$ is a strong solution to a planning problem of the form

$$Y_j(b_{\pi(1)}) \otimes V(b_{\pi(2)}, .., b_{\pi(n)}) \Longrightarrow \widetilde{Z}^{Sym}(b_1, b_2, .., b_n), \tag{32}$$

which implies, in particular, that each T_j can be completed to a game tree $\mathcal{T}_j$ so that $\mathcal{T}_j$ ends in $\{Z_1, .., Z_k\}$.

Replacing each T_j by its completion $\mathcal{T}_j$ yields the desired completion $\mathcal{T}$ of the original game tree T.

Thus each of the 'intermediate' plans $\mathcal{P}_U$ is proved to be a strong solution to the corresponding planning problem (30).

The proof of Theorem 4.2 is completed by stating that the 'big' plan $\mathcal{P}$ constructed above is of the form $\mathcal{P}_W$, and thereby plan $\mathcal{P}$ is a strong solution to the original planning problem (27). ■

Comment 4.2. Notice that

(a) the height of the plan $\mathcal{P}$ assembled within the proof of Theorem 4.2 is bounded by the height of a given HLL-canonical derivation D, and,
(b) more important, the size of $\mathcal{P}$ is bounded by the number of *different* sequents in D.

The *control chart* of $\mathcal{P}$ can be easily obtained by reversing all arrows in D. (Cf. Figs. 2 and 3)

Comment 4.3. Theorem 4.2 provides an exact correspondence between proofs for generic Horn sequents and strong solutions to real planning problems in one direct step:

"generic proofs $\overset{\text{direct}}{\Longrightarrow}$ real plans". (33)

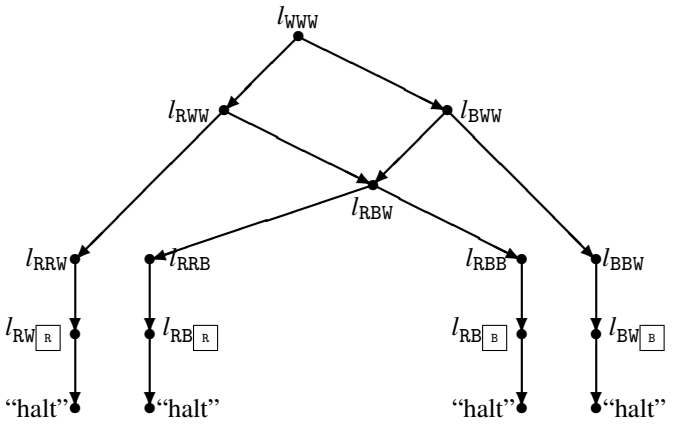


Fig. 3. The control chart for a plan in Example 4.2, with $N=3$, $k=2$.

Bearing in mind Theorem 4.1, it seems more natural to make such a large step in a number of smaller steps like

"generic proofs $\Longrightarrow$ real proofs $\Longrightarrow$ real plans". (34)

Unfortunately, for many planning instances, the corresponding real proofs happen to be of *exponential size* (see, for instance, Comment 4.5 for Example 4.2). As a result, we could not have obtained polynomial strong solutions if we had followed the strategy of line (34).

4.1. Strong planning in deterministic domains

Now we can explain why our technique provides automatically a correct solution to Example 1.11.

Corollary 4.1. *Within Example 1.11, the 'real' planning problem*

$$\mathtt{In}_{N,k}(h_1, h_2, .., h_k,\ b_1, b_2, .., b_N) \vdash \mathtt{Goal}_N(b_1, b_2, .., b_N)$$

can be fully sorted out in terms of the 'generic' planning problem

$$\mathtt{In}_{N,k}(h, h, .., h,\ b, b, .., b) \vdash \mathtt{Goal}_N(b, b, .., b).$$

Proof. This follows immediately from Theorem 4.2, since each of the action specifications there is 'ball'-monadic and 'grip'-monadic. ■

Example 4.1. A robot has k grips. It can carry a ball in each. There are N_1 balls in one room, and N_2 balls in another. The goal is to exchange these two heaps of balls.

Notwithstanding the confusion in Example 1.12 with $N_2=N_1=1$, our machinery is still capable of providing `polytime` solutions to Example 4.1.

The problem with Example 4.1, as well as with Example 1.12, is that the goal is not symmetric with respect to all N_1+N_2 variables together.

But we can break down these variables into two groups: u_1, .., u_{N_1}, representing the first N_1 balls, and v_1, .., v_{N_2}, representing the last N_2 balls. It is clear that our goal *is* symmetrical with respect to u_1, .., u_{N_1}, and our goal *is* symmetrical with respect to v_1, .., v_{N_2}.

We will think about u_1, .., u_{N_1} as copies of a 'generic ball', say u, and we take v_1, .., v_{N_2} as copies of a 'generic ball' v. Now it remains to apply Theorem 4.2 and 5.2 but to these *two* 'generic balls' u and v. ■

Comment 4.4. Example 4.1 suggests an idea of a general strategy to detect the symmetries we can use for making plans of polynomial size:

(a) Check that all *actions* can be specified by axioms that are monadic with respect to some sort τ. (See Definition 4.4)
(b) Find that the *goal* in question is symmetrical with respect a group of variables of sort τ.

4.2. Strong planning under uncertainty: "Rouge ou Noir"

We illustrate the machinery of our approach with a *knowledge acquisition* game 'Robot against Nature' in Example 4.2, which combines combinatorially exploded features of a deterministic "Gripper" [1] and a non-deterministic "Socks".

Example 4.2. *"Rouge ou Noir"*: A robot deals with a heap of N balls wrapped with paper. Each ball is either red or black; the robot can unwrap any ball revealing its colour. There are k containers; each of them may contain only one ball. The robot can put a red (black) ball into a container and mark the container red (black, respectively).

Initially, all containers are empty. The goal is to make sure the robot fills all containers with balls of one and the same colour.

Compared to the deterministic domains discussed above (which can be conceived of as specific versions of the Pigeonhole Principle), the non-deterministic Example 4.2 can be thought of as a simplified version of Ramsey's Theorem [32], a generalization of the Pigeonhole Principle.[5]

It is readily seen that there is a winning strategy in *"Rouge ou Noir"* if and only if $N \geq 2k-1$.

The full picture is more subtle (see also Example 6.1):

$N \leq k-1$	$k \leq N \leq 2k-2$	$N \geq 2k-1$
No solution	Weak solution	Strong solution

We introduce the following predicates:

(a) $\mathtt{Wrap}(z)$ stands for *"ball z is wrapped with paper"*,
(b) $\mathtt{Red}(z)$ means *"an unwrapped ball z has turned out to be red"*,
(c) $\mathtt{Black}(z)$ means *"an unwrapped ball z has turned out to be black"*,
(d) $\mathtt{Empty}(y)$ stands for *"container y is empty"*,
(e) $\mathtt{Place}_{\mathtt{R}}(z, y)$ means *"a red ball z is put into container y, with y being marked red"*, and
(f) $\mathtt{Place}_{\mathtt{B}}(z, y)$ means *"a black ball z is put into container y, with y being marked black"*.

The 'learn' action $\mathtt{learn}(z)$:

"Unwrap ball z revealing its color",

is formalized as

$$\mathtt{Wrap}(z) \vdash (\mathtt{Red}(z) \oplus \mathtt{Black}(z)); \tag{35}$$

the *disjunctive* $\oplus$-form of which emphasizes *uncertainty* about the effects of such an unwrapping action: the robot does not know, *in advance*, which colour the ball chosen will be.

The 'put-and-mark' actions $\mathtt{put}_{\mathtt{R}}(z, y)$ and $\mathtt{put}_{\mathtt{B}}(z, y)$:

"Put ball z of the known colour into container y, and mark y with this colour",

are formalized, respectively, as

$$(\mathtt{Red}(z) \otimes \mathtt{Empty}(y)) \vdash \mathtt{Place}_{\mathtt{R}}(z, y) \tag{36}$$

$$(\mathtt{Black}(z) \otimes \mathtt{Empty}(y)) \vdash \mathtt{Place}_{\mathtt{B}}(z, y) \tag{37}$$

For N balls, say $b_1, b_2, .., b_N$, and k containers, say $c_1, c_2, .., c_k$, the initial configuration is defined as

$$\mathtt{In}_{N,k}(b_1, .., b_N, c_1, .., c_k) := \bigotimes_{i=1}^{N} \mathtt{Wrap}(b_i) \otimes \bigotimes_{j=1}^{k} \mathtt{Empty}(c_j),$$

[5] Ramsey's Theorem [32] declares, in particular, that *"For any positive integer k, there exists a positive integer N such that no matter how the complete graph with N nodes is coloured in red and black, it will contain a red complete subgraph with k nodes or a black complete subgraph with k nodes."*

the goal is represented as a formula of the form

$$\mathtt{Goal}_{N,k}(b_1,..,b_N,c_1,..,c_k) := \bigoplus_{(i_1,..,i_k)} \bigotimes_{j=1}^{k} \mathtt{Place}_{\mathtt{R}}(b_{i_j},c_j) \;\oplus\; \bigoplus_{(i_1,..,i_k)} \bigotimes_{j=1}^{k} \mathtt{Place}_{\mathtt{B}}(b_{i_j},c_j)$$

where $(i_1,..,i_k)$ ranges over all permutations of $\{1,..,N\}$ taken k at a time.
The 'real' planning problem in Example 4.2

$$\mathtt{In}_{N,k}(b_1,b_2,..,b_N,c_1,c_2,..,c_k) \Rightarrow \mathtt{Goal}_{N,k}(b_1,b_2,..,b_N,c_1,c_2,..,c_k). \tag{38}$$

is to achieve the goal within the "worst-case scenario": a plan we are looking for must envisage *all* possible reactions of Nature on the road from the *initial position* to one of the *final positions*.

Any solution to this *strong planning* problem seems to be at least *exponential*, since the number of the corresponding 'red-black' distributions we might have met with is $\Omega(2^N)$.
Our *erasing individuality* trick yields the *mock* 'generic' planning problem

$$\mathtt{In}_{N,k}(b,b,..,b,c,c,..,c) \Rightarrow \mathtt{Goal}_{N,k}(b,b,..,b,c,c,..,c), \tag{39}$$

with $\mathtt{In}_{N,k}(b,b,..,b,c,c,..,c)$ being of the form

$$(\underbrace{\mathtt{Wrap}(b)\otimes\cdots\otimes\mathtt{Wrap}(b)}_{N\ times})\otimes(\underbrace{\mathtt{Empty}(c)\otimes\cdots\otimes\mathtt{Empty}(c)}_{k\ times})$$

and $\mathtt{Goal}_{N,k}(b,b,..,b,c,c,..,c)$ being of the form

$$(\underbrace{\mathtt{Place}_{\mathtt{R}}(b,c)\otimes\cdots\otimes\mathtt{Place}_{\mathtt{R}}(b,c)}_{k\ times})\oplus(\underbrace{\mathtt{Place}_{\mathtt{B}}(b,c)\otimes\cdots\otimes\mathtt{Place}_{\mathtt{B}}(b,c)}_{k\ times}).$$

4.2.1. Searching a 'generic' proof

To resolve the generic planning problem (39), first, we have to prove the following sequent:

$$\mathtt{In}_{N,k}(b,b,..,b,c,c,..,c) \vdash \mathtt{Goal}_{N,k}(b,b,..,b,c,c,..,c). \tag{40}$$

The number of all 'generic' configurations in question is $O(N^2k^2)$. Hence, an HLL-canonical derivation for (40), if there is one, can be assembled in *polytime* in a bottom-up manner (a similar technique is used in Theorem 5.2).

For the case where $N=3$, $k=2$, an HLL-canonical derivation for sequent (40) is shown in Fig. 2. Here we abbreviate certain LL-monomials, namely, W stands for $\mathtt{Wrap}(b)$, R for $\mathtt{Red}(b)$, B for $\mathtt{Black}(b)$, and ε for $\mathtt{Empty}(c)$, $\boxed{\mathtt{B}}$ for $\mathtt{Place}_{\mathtt{B}}(b,c)$, and $\boxed{\mathtt{R}}$ for $\mathtt{Place}_{\mathtt{R}}(b,c)$. The goal $\mathtt{Goal}_{3,2}(b,b,b,c,c)$ is abbreviated as $\boxed{\mathtt{B}}\boxed{\mathtt{B}}\oplus\boxed{\mathtt{R}}\boxed{\mathtt{R}}$.

4.2.2. Converting to a strong 'real' plan

For the sake of clarity, first we plot a *control chart* of a plan $\mathcal{P}$ we are looking for by reversing arrows in the derivation found. For $N=3$, $k=2$, the result is shown in Fig. 3.

The plan $\mathcal{P}$ itself is assembled by collecting all commands associated with nodes of the derivation (we merge the halting commands for brevity):

$l_{\mathtt{WWW}}$: Given a state of the form $\mathtt{Wrap}(d)\otimes V'$ where $d\in\{b_1,b_2,b_3\}$,
apply $\mathtt{learn}(d)$;
upon getting response $\mathtt{Red}(d)$, go to $l_{\mathtt{RWW}}$,
upon getting response $\mathtt{Black}(d)$, go to $l_{\mathtt{BWW}}$;
$l_{\mathtt{RWW}}$: Given a state of the form $\mathtt{Wrap}(d)\otimes V'$ where $d\in\{b_1,b_2,b_3\}$,
apply $\mathtt{learn}(d)$;
upon getting response $\mathtt{Red}(d)$, go to $l_{\mathtt{RRW}}$,
upon getting response $\mathtt{Black}(d)$, go to $l_{\mathtt{RBW}}$;
$l_{\mathtt{BWW}}$: Given a state of the form $\mathtt{Wrap}(d)\otimes V'$ where $d\in\{b_1,b_2,b_3\}$,
apply $\mathtt{learn}(d)$;
upon getting response $\mathtt{Red}(d)$, go to $l_{\mathtt{RBW}}$,
upon getting response $\mathtt{Black}(d)$, go to $l_{\mathtt{BBW}}$;

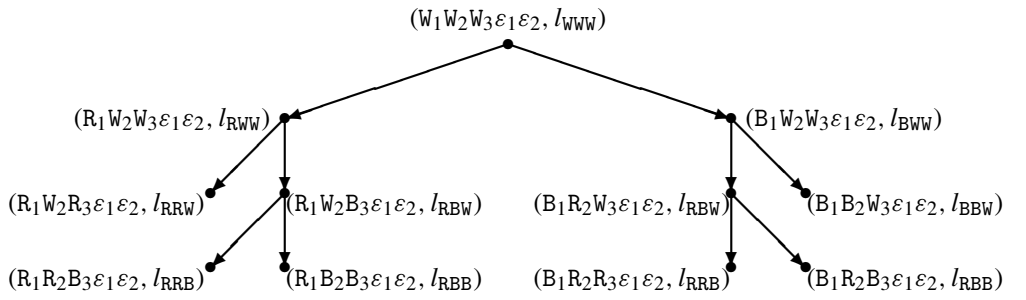


Fig. 4. A partial game tree in Example 4.2, with $N=3, k=2$. Here $\mathtt{W}_i$ stands for $\mathtt{Wrap}(b_i)$, $\mathtt{R}_i$ for $\mathtt{Red}(b_i)$, $\mathtt{B}_i$ for $\mathtt{Black}(b_i)$, and ε_j for $\mathtt{Empty}(c_j)$.

$l_{\mathtt{RBW}}$: Given a state of the form $\mathtt{Wrap}(d) \otimes V'$ where $d \in \{b_1, b_2, b_3\}$,
apply $\mathtt{learn}(d)$;
upon getting response $\mathtt{Red}(d)$, go to $l_{\mathtt{RRB}}$,
upon getting response $\mathtt{Black}(d)$, go to $l_{\mathtt{RBB}}$;

$l_{\mathtt{RRW}}$: Given a state of the form $\mathtt{Red}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{R}}(d, e)$; go to $l_{\mathtt{RW}\boxed{\mathtt{R}}}$;

$l_{\mathtt{RW}\boxed{\mathtt{R}}}$: Given a state of the form $\mathtt{Red}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{R}}(d, e)$; go to l_0;

$l_{\mathtt{RRB}}$: Given a state of the form $\mathtt{Red}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{R}}(d, e)$; go to $l_{\mathtt{RB}\boxed{\mathtt{R}}}$;

$l_{\mathtt{RB}\boxed{\mathtt{R}}}$: Given a state of the form $\mathtt{Red}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{R}}(d, e)$; go to l_0;

$l_{\mathtt{RBB}}$: Given a state of the form $\mathtt{Black}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{B}}(d, e)$; go to $l_{\mathtt{RB}\boxed{\mathtt{B}}}$;

$l_{\mathtt{RB}\boxed{\mathtt{B}}}$: Given a state of the form $\mathtt{Black}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{B}}(d, e)$; go to l_0;

$l_{\mathtt{BBW}}$: Given a state of the form $\mathtt{Black}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{B}}(d, e)$; go to $l_{\mathtt{BW}\boxed{\mathtt{B}}}$;

$l_{\mathtt{BW}\boxed{\mathtt{B}}}$: Given a state of the form $\mathtt{Black}(d) \otimes \mathtt{Empty}(e) \otimes V'$
where $d \in \{b_1, b_2, b_3\}$, and $e \in \{c_1, c_2\}$,
apply $\mathtt{Place}_{\mathtt{B}}(d, e)$; go to l_0;

l_0: halt;

The command labelled with $l_{\mathtt{WWW}}$ is designated to be the initial command.
For the initial position

$$\mathtt{Wrap}(b_1) \otimes \mathtt{Wrap}(b_2) \otimes \mathtt{Wrap}(b_3) \otimes \mathtt{Empty}(c_1) \otimes \mathtt{Empty}(c_2),$$

one of the partial game trees developed according to $\mathcal{P}$ is drawn in Fig. 4.

Comment 4.5. To compare our folded plans with tree-like plans, notice that any successful game tree for Example 4.2 must involve at least 2^k distinct positions. Therefore, for any HLL-canonical derivation D for the sequent representing our 'real' planning problem (38):

$$\mathtt{In}_{N,k}(b_1, b_2, .., b_N, c_1, c_2, .., c_k) \vdash \mathtt{Goal}_{N,k}(b_1, b_2, .., b_N, c_1, c_2, .., c_k),$$

the size of D is $\Omega(2^k)$. The effect is that, in particular, any tree-like plan is of the exponential size $\Omega(2^k)$.

5. Complexity

In this section we address complexity issues related to practical applications of our approach.

5.1. Decidability in general

A general case of non-deterministic domains has been investigated in KanovichVauzeilles [22]. It has been shown there that the case of so-called 'exact goals' is expressed in linear logic proper, whereas the case of so-called 'partial goals' is expressed in affine logic.

Since the corresponding fragment of linear logic is undecidable [20], the general case of 'exact goals' is undecidable, as well. Fortunately, propositional affine logic turns out to be decidable [25,26]. And we take advantage of our linear logic formalism to show decidability of the general planning problem under uncertainty dealing with 'partial goals'.

Corollary 5.1 (*Decidability*). *There is an algorithm applicable to the following pair of inputs:*

(i) *to any theory T over a finite signature such that the set of its proper axioms $\mathtt{Ax}_T$ is a finite set of disjunctive Horn sequents, and,*
(ii) *to any closed LL-monomials W, Z_1,..,Z_k,*

which:

(a) *determines whether there is a strong plan that is based on $\mathtt{Ax}_T$ leading from W to $\{Z_1, .., Z_k\}$, and,*
(b) *if the answer is positive, produces such a plan.*

Proof. By Theorem 3.1 we express the planning problem considered as a derivability problem in affine logic, which is decidable [25,26]. ■

5.2. Weighted balance ⟶ EXPTIME planning

The only drawback of the proof in Corollary 5.1 is that it yields a very complex algorithm, since the related decision procedure in [25,26] is based on the parallel execution of two competitive complex processes: one process is looking for an affine logic derivation for a given sequent, while the another process is checking finite 'phase models' aiming at a counter-example to this sequent.

Here, with the help of a 'mixed balance', we design much more feasible planning algorithms for a reasonably wide class of AI robot systems (cf. [22]).

Definition 5.1. Suppose that sorts τ_1, τ_2,..., τ_ℓ are fixed.
Given an LL-monomial X, we define its *mixed weight* $\omega_{\tau_1\tau_2..\tau_\ell}(X)$ as follows:

$$\omega_{\tau_1\tau_2..\tau_\ell}(X) \;:=\; M + \#_{\tau_1}(X) + \#_{\tau_2}(X) + \cdots + \#_{\tau_\ell}(X)$$

where M is the number of occurrences of atomic formulas in X that do not contain an occurrence of a variable of any of the sorts τ_1, τ_2, ..., τ_ℓ.
Notice that the degree of X (that is $\otimes$-rank in terms of [22]) is less or equal to $\omega_{\tau_1\tau_2..\tau_\ell}(X)$.

E.g., for $Y = (\mathtt{Room}(x) \otimes \mathtt{Hold}(y, z))$, we get: $\omega_{ball,grip}(Y) = 3$;
for $X = (\mathtt{Room}(x) \otimes \mathtt{Empty}(y) \otimes \mathtt{Floor}(x, z))$, we get: $\omega_{ball,grip}(X) = 3$.

Definition 5.2. Let $\mathtt{Ax}_T$ consist of disjunctive Horn sequents. We will say that $\mathtt{Ax}_T$ is *well-balanced* with respect to given sorts τ_1, ..., τ_ℓ ($\ell \geq 0$) if for any sequent $X \vdash (Y_1 \oplus \cdots \oplus Y_m)$ taken from $\mathtt{Ax}_T$,

$$\omega_{\tau_1..\tau_\ell}(Y_1) \leq \omega_{\tau_1..\tau_\ell}(X) \quad \ldots, \quad \omega_{\tau_1..\tau_\ell}(Y_m) \leq \omega_{\tau_1..\tau_\ell}(X).$$

For instance, the system of axioms in Example 1.11 turns out to be *well-balanced* with respect to sorts *ball* and *grip*.

(As compared to [22], some axioms in Example 1.11 are not well-balanced with respect to the degree of LL-monomials, and, hence, Example 1.11 is not within the direct reach of [22, Theorem 5.2]).

Theorem 5.1. *Let $\mathtt{Ax}_T$ be well-balanced with respect to some sorts τ_1, ..., τ_ℓ.*

Then we can construct an algorithm α running in exponential time, so that for any closed LL-monomials W, Z_1,..,Z_k, algorithm α

(a) *determines whether there is a strong plan that is based on $\mathtt{Ax}_T$ and leads from W to $\{Z_1, .., Z_k\}$, and,*
(b) *if the answer is positive, produces such a strong plan.*

Proof. Take the proof of Theorem 5.2 in [22], and replace the degree of X with $\omega_{\tau_1\tau_2..\tau_\ell}(X)$. ■

Comment 5.1. Recalling that *"Towers of Hanoi"* in Example 1.4 requires plans of exponential length, we see that Theorem 5.1 gives the best upper bound for time complexity of planning in some deterministic domains.

5.3. Monadic & balanced ⟶ polytime planning

Lemma 5.1. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents. In addition, let $\mathtt{Ax}_T$ be well-balanced with respect to some sorts τ_1, ..., τ_ℓ.*

Let z_1, z_2,.., z_n be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial.
Suppose that D is a derivation in the HLL-theory T developed for the sequent

$$W(z, z, .., z) \vdash \widetilde{Z}(z, z, .., z). \tag{41}$$

Then every sequent occurring in D is of the form $U(z, .., z) \vdash \widetilde{Z}(z, .., z)$, for some LL-monomial $U(z_1, z_2, .., z_n)$ in which each of the z_i has at most one occurrence and for which:

$$\omega_{\tau_1\tau_2..\tau_\ell}(U(z_1, z_2, .., z_n)) \leq \omega_{\tau_1\tau_2..\tau_\ell}(W(z_1, z_2, .., z_n)).$$

Proof. Similar to Lemma 4.1. ■

Theorem 5.2. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents, and let $\mathtt{Ax}_T$ be well-balanced with respect to some sorts τ_1, ..., τ_ℓ.*

Let z_1,z_2,..,z_n be variables of sort τ, and let $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be a symmetrical LL-polynomial such that $\#_{z_i}(\widetilde{Z}) \leq 1$ for each z_i.

Let b_1,b_2,..,b_n be constants of sort τ having no occurrence in $\mathtt{Ax}_T$, W, and $\widetilde{Z}$.

Then we can determine in polynomial time (with respect to n) whether there is a strong plan for the planning problem

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}(b_1, b_2, .., b_n) \tag{42}$$

and, if the answer is positive, produce such a strong plan in polynomial time.

Proof. According to Theorem 4.2, the original planning problem is equivalent to HLL-derivability of a 'generic' sequent of the form

$$W(b, b, .., b) \vdash \widetilde{Z}(b, b, .., b) \tag{43}$$

Take $m = \omega_{\tau_1\tau_2..\tau_\ell}(W(z_1, z_2, .., z_n))$.

Because of Lemma 5.1, we can confine ourselves to HLL-canonical derivations D where each of the sequents is of the form

$$U(b, .., b) \vdash \widetilde{Z}(b, .., b)$$

for some LL-monomial $U(z_1, z_2, .., z_n)$ in which each of the z_i has at most one occurrence and for which:

$$\text{degree of } U \leq \omega_{\tau_1\tau_2..\tau_\ell}(U(z_1, z_2, .., z_n)) \leq m.$$

The number of all $U(b, .., b)$ whose degree does not exceed m is bounded by some polynomial p; the degree of p is determined by the number of predicate symbols, their arity, and the number of constants in $\mathtt{Ax}_T$, $W(b, .., b)$, and $\widetilde{Z}(b, .., b)$.

Applying a bottom-up technique (see details, for instance, in [22, Corollary 5.1]), we can construct in *polytime* an HLL-canonical proof D without repetitions for sequent (43). The number of *distinct* sequents in D turns out to be bounded by the above polynomial p, at least.

Theorem 4.2 transforms D into a plan $\mathcal{P}$ that is a strong solution to the original planning problem

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}(b_1, b_2, .., b_n).$$

Comment 4.2 guarantees that $\mathcal{P}$ is of polynomial size. ■

6. Weak planning under uncertainty

According to Definition 3.4, the weak planning problem is related to the chance of fortunate events.

To illustrate the difference between strong planning and weak planning, we invoke the same Example 4.2 but with specific parameters N and k.

Example 6.1. *"Rouge ou Noir"*: A robot deals with a heap of N balls wrapped with paper. Each ball is either red or black; the robot can unwrap any ball revealing its colour. There are k containers; each of them may contain only one ball. It is assumed that

$$k \leq N \leq 2k-2.$$

The robot can put a red (black) ball into a container and mark the container red (black, respectively).

Initially, all containers are empty. The goal is to make sure the robot fills all containers with balls of one and the same color.

There is *no* strong solution to Example 6.1, since the robot would have failed if Nature had entrapped the robot by preparing no more than $k-1$ red balls and no more than $k-1$ black balls.

On the other hand, the planning situation is not absolutely hopeless: Expecting the most optimistic outcomes, say "each ball is red", the robot has a lucky chance to accomplish its task. Hence, a weak solution to the planning problem still exists. ■

We prove that the weak planning problem can be fully handled within the pure deterministic paradigm.

Definition 6.1. Given a theory T whose $\mathtt{Ax}_T$ consists of disjunctive Horn sequents, its *weakened* version $T^{\mathcal{W}}$ is obtained by the following replacement procedure.

Each of the disjunctive Horn sequents from $\mathtt{Ax}_T$

$$X \vdash (Y_1 \oplus \cdots \oplus Y_m) \tag{44}$$

is replaced with m pure Horn sequents of the form:

$$\begin{array}{l} X \vdash Y_1 \\ \cdots \\ X \vdash Y_m \end{array} \tag{45}$$

Thus the 'learn' action (35) with non-deterministic effects:

$$\mathtt{Wrap}(z) \vdash (\mathtt{Red}(z) \oplus \mathtt{Black}(z)),$$

will be 'weakened' into two actions with deterministic effects:

(1) $\mathtt{Wrap}(z) \vdash \mathtt{Red}(z)$
(2) $\mathtt{Wrap}(z) \vdash \mathtt{Black}(z)$

We show that the weak plans within a theory T are exactly the strong plans within its weakened theory $T^{\mathcal{W}}$.

Theorem 6.1. *Given a planning problem $W \Rightarrow \widetilde{Z}$ within a theory T, any plan $\mathcal{P}$ that is a strong solution to $W \Rightarrow \widetilde{Z}$ within the theory $T^{\mathcal{W}}$ can be straightforwardly rewritten as a weak solution to $W \Rightarrow \widetilde{Z}$ within T; and vice versa.*

Proof. The most interesting direction is to show how weak plans for the original problem can be extracted from strong solutions within the weakened theory $T^{\mathcal{W}}$. (The converse is justified by the same token.)

The weakened theory $T^{\mathcal{W}}$ invokes only pure Horn sequents to specify its actions. Therefore, the control chart of any strong plan $\mathcal{P}$ within $T^{\mathcal{W}}$ must be a *chain*. In particular, in each of its commands of the form (18) with parameters $d_1,..,d_\ell$ we can contract their domains $D_1,..,D_\ell$ to singletons. The effect is that each of the non-halting commands in this strong plan $\mathcal{P}$ within $T^{\mathcal{W}}$ is of the form:

$$\boxed{\begin{array}{l} l\text{: apply the action specified with } X \vdash Y_i, \\ \qquad \text{which is an instance of a sequent from } \mathtt{Ax}_{T^{\mathcal{W}}}; \\ \text{go to the command labelled with } l'. \end{array}} \tag{46}$$

To obtain a weak solution to the original planning problem within T, we replace the body of (46) with a body from $\mathtt{Ax}_T$ in the following way.

Let $X \vdash Y_i$ in (46) come up from a sequent of the form

$$X \vdash (Y_1 \oplus \cdots \oplus Y_i \oplus \cdots \oplus Y_m).$$

Then we replace each of these (46) by:

$$\boxed{\begin{array}{l} l\text{: apply the action specified with } X \vdash (Y_1 \oplus \cdots \oplus Y_i \oplus \cdots \oplus Y_m), \\ \qquad \text{which is an instance of a sequent from } \mathtt{Ax}_T; \\ \text{upon getting response } Y_1\text{, go to a halting command,} \\ \ldots\ldots \\ \text{upon getting response } Y_i\text{, go to } l', \\ \ldots\ldots \\ \text{upon getting response } Y_m\text{, go to a halting command.} \end{array}} \tag{47}$$

As a result, we obtain a plan, say $\mathcal{P}'$, such that its control chart is almost the same as the control chart of $\mathcal{P}$: the only difference is that a number of additional arrows leading to a halting command.

It is readily seen that $\mathcal{P}'$ provides a game tree having at least one successful branch, and, hence, $\mathcal{P}'$ is a weak solution to the original planning problem within T. ■

Corollary 6.1. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents.*

Let $z_1,z_2,..,z_n$ be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial such that for each z_i, $\#_{z_i}(\widetilde{Z}) \leq 1$.

Let b, b_1, b_2,.., b_n be constants of sort τ having no occurrence in $\mathtt{Ax}_T$, W, and $\widetilde{Z}$.

Then every cut-free affine logic proof within the weakened theory $T^{\mathcal{W}}$ for a sequent of the form

$$W(b, b, .., b) \vdash \widetilde{Z}(b, b, .., b)$$

which deals with one *'generic object' b, can be converted (in* `polytime`*) into a weak solution to the original planning problem specified in theory T as*

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}^{Sym}(b_1, b_2, .., b_n),$$

which deals with n 'real objects'.

Proof. This follows immediately from Theorems 6.1 and 4.2. ■

Corollary 6.2. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic disjunctive Horn sequents, and let $\mathtt{Ax}_T$ be well-balanced with respect to some sorts τ_1, ..., τ_ℓ.*

Let z_1,z_2,..,z_n be variables of sort τ, and $W(z_1, z_2, .., z_n)$ be an LL-monomial in which each of the z_i has exactly one occurrence, and $\widetilde{Z}(z_1, z_2, .., z_n)$ be an LL-polynomial such that $\#_{z_i}(\widetilde{Z}) \leq 1$ for each z_i.

Let b_1,b_2,..,b_n be constants of sort τ having no occurrence in $\mathtt{Ax}_T$, W, and $\widetilde{Z}$.

Then we can determine in polynomial time (with respect to n) whether there is a weak plan for the planning problem

$$W(b_1, b_2, .., b_n) \Longrightarrow \widetilde{Z}(b_1, b_2, .., b_n) \tag{48}$$

and, if the answer is positive, produce such a weak plan in polynomial time.

Proof. This follows immediately from Theorems 6.1 and 5.2. ■

7. Concluding remarks

7.1. A comparison with a bisimulation approach

In our general theorems such as Theorem 4.1 the symmetry of a planning goal plays an important role. Such a constraint can be significantly relaxed in the case of actions with only deterministic effects.

Theorem 7.1. *For a given sort τ, let $\mathtt{Ax}_T$ consist only of τ-monadic pure Horn sequents.*

Then for any variables z, z_1,z_2,..,z_n of sort τ the following holds:

Whatever LL-monomial $W(z_1, z_2, .., z_n)$ in which each of the z_i has exactly one occurrence, and LL-polynomial $\widetilde{Z}(z_1, z_2, .., z_n)$ such that $\#_{z_i}(\widetilde{Z}) \leq 1$ for each z_i, we take, if a sequent of the form

$$W(z, z, .., z) \vdash \widetilde{Z}(z, z, .., z) \tag{49}$$

is derivable from $\mathtt{Ax}_T$ by the rules of affine logic, then for some permutation π, a sequent of the form

$$W(z_1, z_2, .., z_n) \vdash \widetilde{Z}(z_{\pi(1)}, z_{\pi(2)}, .., z_{\pi(n)}) \tag{50}$$

is also derivable from $\mathtt{Ax}_T$ by the rules of affine logic.

Proof. Similar to Theorem 4.1 ■

Theorem 7.1 suggests an idea of a certain *bisimulation* equivalence induced by symmetry of the system at hand.

Definition 7.1. Namely, for fixed variables z_1,z_2,..,z_n of a sort τ, we will consider LL-monomials

$$U(z_1, z_2, .., z_n)$$

and (π is a permutation)

$$U(z_{\pi(1)}, z_{\pi(2)}, .., z_{\pi(n)})$$

as *equivalent;* here $U(z_1, z_2, .., z_n)$ is an LL-monomial such that $\#_{z_i}(U) \leq 1$ for each of these z_i.

The main hypothesis in Theorem 7.1 — that $\mathtt{Ax}_T$ consists only of τ-monadic pure Horn sequents, automatically guarantees *bisimilarity* of the above equivalence, namely:

Proposition 7.1. *Let LL-monomials U and U' be equivalent (with respect to variables z_1,z_2,..,z_n of a sort τ), and let U be transformed into an LL-monomial V by means of an instance of a τ-monadic pure Horn sequent*

$$X(z) \vdash Y(z).$$

Then there is an LL-monomial V' such that V' is equivalent to V, and U' can be transformed into V' by means of an instance of the same pure Horn sequent.

Comment 7.1. There have been certain attempts to automatically detect symmetries and explore them in other areas such as SAT checking [31]. Their approach to problem solving is to start with an *exponential* state space and then partition it with respect to the symmetry detected.

But the symmetry by itself does not help to reduce the number of the variables involved in making plans, since any representative for an equivalence class is of the form

$$U(z_1, z_2, .., z_n),$$

and still deals with a variety of variables z_1,z_2,..,z_n.

Our approach to the planning problem is quite opposite. It was our trick that an LL-monomial in one variable z of the form

$$U(z, z, .., z)$$

served as a 'generic' representative for the corresponding equivalence class. In that way, we started from a *small* 'generic' space, and then produced efficient solutions to the real planning problem at hand by exploring this 'generic' space only, without touching the combinatorially exploded real space at all.

In fact, a real plan $\mathcal{P}$ provided by Theorems 7.1 and 4.2 can be conceived of as a representative of some abstract path over the corresponding equivalence classes. ■

It should be pointed out that the above line of reasoning works properly only for deterministic domains.

As for the general case of actions with non-deterministic effects, both Theorem 7.1 and the related bisimulation generally fail.

Example 7.1. Let $\mathtt{Ax}_T$ consist of the following axioms:

$$\begin{cases} A \otimes P(z) \vdash (B \otimes L(z)) \oplus (C \otimes R(z)) \\ B \otimes P(z) \vdash D \otimes R(z) \\ C \otimes P(z) \vdash D \otimes L(z) \end{cases} \tag{51}$$

Then a 'generic' sequent of the form

$$A \otimes P(b) \otimes P(b) \vdash D \otimes L(b) \otimes R(b)$$

is derivable from (51), but neither

$$A \otimes P(b_1) \otimes P(b_2) \vdash D \otimes L(b_1) \otimes R(b_2)$$

nor

$$A \otimes P(b_1) \otimes P(b_2) \vdash D \otimes L(b_2) \otimes R(b_1)$$

is derivable from (51) by affine logic rules.

Indeed, in full accordance with Theorem 4.1, the sequent with a completely symmetrical goal:

$$A \otimes P(b_1) \otimes P(b_2) \vdash (D \otimes L(b_1) \otimes R(b_2)) \oplus (D \otimes L(b_2) \otimes R(b_1)),$$

is derived from (51) by linear logic rules.

Corollary 7.1. *In fact, Example 7.1 reflects a deeper borderline related to the so-called* disjunctive property.

(a) *For the case where each action has a deterministic effect, we can prove that any planning problem of the form*

$$W \Rightarrow (Z_1 \oplus Z_2)$$

has a strong solution if and only if one of the planning problems:

$$W \Rightarrow Z_1$$

or

$$W \Rightarrow Z_2,$$

has a strong solution.

(b) *On the other hand, Example 7.1 gives a strongly solvable planning problem of the form* $W \Rightarrow (Z_1 \oplus Z_2)$ *such that a planning problem of the form* $W \Rightarrow Z_1$ *has no strong solution, and neither has a planning problem of the form* $W \Rightarrow Z_2$.

Proof. (a) Since we use here only pure Horn sequents, the control chart of any strong solution $\mathcal{P}$ to $W \Rightarrow (Z_1 \oplus Z_2)$ is a *chain*. This allows us to select an appropriate Z_i so that $\mathcal{P}$ will be a strong solution to $W \Rightarrow Z_i$. ■

7.2. Yet another formalism ...

One could say that some examples given in the paper (e.g. Example 1.11) could be reformulated and solved in several logical planning formalisms, such as the situation calculus, functional STRIPS [14], etc. A number of works exploit symmetries, for instance, for deterministic planning [12,13] or for SAT representations of transition systems [33].

What arguments could we offer about the superiority of linear logic as a modelling formalism for the planning domain considered here (save its being novel compared to the others) ?

(A) First, linear logic provides a uniform approach to specifying deterministic and non-deterministic domains in *user terms*, without radical reformulation of the original problem.

On top of that, the linear logic approach provides a direct and clear *correspondence between proofs and plans*.

(B) As for the 'symmetrical' domains with numerous but identical elements, we employ the ability of linear logic to *reason about multisets*, which in this instance are created by identifying several distinct objects as being functionally equivalent for the problem at hand (think of a number of balls, each of which must be moved to some new location — the balls are distinct, but are functionally equivalent for the problem within Example 1.11).

We have established a clear syntactic condition (see Definition 4.4) that allows us to show that solving a generic planning problem where there is only *one generic object*, directly implies a solution to the original real planning problem over *several real objects*, the isomorphic 'copies' of the generic object.

Moreover, this correspondence also guarantees to produce a real solution that works in *polynomial time*.

Thus linear logic allows us to *circumvent* the combinatorial explosion in real state spaces instead of *straight overcoming* the real spaces exploded.

It seems very problematic to reformulate and justify these results within traditional logic formalisms.

(C) No changes in the user structure of the planning problem is made by linear logic in order to figure out plans of polynomial size.

Quite the contrary, the linear logic approach *reveals* a much deeper 'generic structure' *behind* the original 'surface structure' for a wide class of deterministic and non-deterministic domains with numerous but identical elements.

This new phenomenon provides *a radical reduction of the number of variables involved in making plans*, which accounts for the surprising fact that, for such systems, the plan existence becomes *polynomial*, while it is known to be generally PSPACE-complete for deterministic domains, and EXPTIME-complete for non-deterministic domains.

(D) In front of actions with non-deterministic effects, plans as winning strategies (that is, trees) are generally of exponential size.

Nevertheless, following linear logic proofs, we can assemble plans in a concise and adequate form of 'acyclic programs with parameterized commands'.

This allows us to explore the difference between situations in which winning strategies are exponential *per se,* and situations in which winning strategies are exponential only because of that the number of different positions that should be mentioned within a game tree happens to be exponential. (See also Comment 3.1) ■

Schematically, our approach to solving planning problems involves the following steps shown in Fig. 5:

(a) 'Generic' Planning Problem $\Longrightarrow$ 'Generic' Plans

On the road from a specification of a generic problem to its solution, we take advantage of the fact that the generic search space is guaranteed to be polynomial. Because of that, we can choose among a wide spectrum of known techniques: direct searching for the shortest paths, theorem proving, BDDs, SAT, etc.

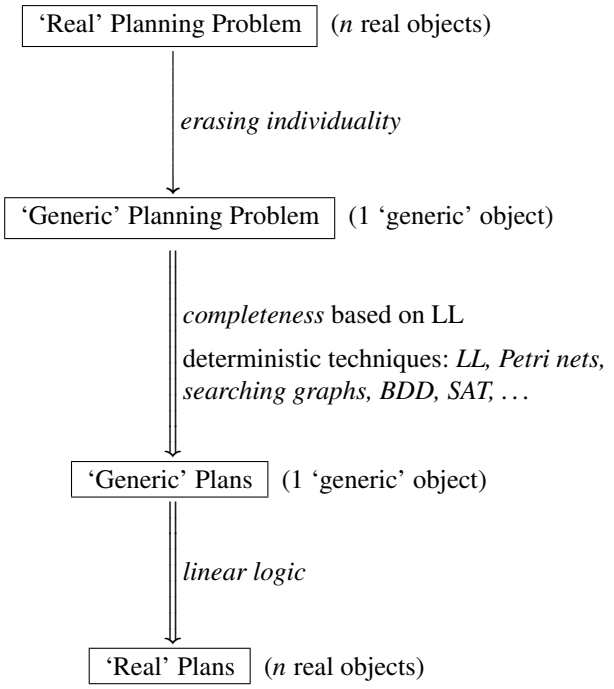


Fig. 5. The 'generic' approach.

Notice that a simple decision procedure is not satisfactory for our purposes: planners should resolve the much more complicated problem of making *actual plans*.

Compared to many existing logic formalisms for planning, the advantage of linear logic here is that it provides a direct and clear correspondence between proofs and plans.

(b) 'Generic' Plans $\Longrightarrow$ 'Real' Plans

The idea that any generic plan in which we abstract away any differences between real objects, can be transformed into a *correct* plan over the real world seems 'orthogonal' to what traditional set-theoretical logical systems are doing.

But such a *concretizing* procedure is easily specified and justified within the linear logic paradigm.

In closing, we have shown that linear logic can automatically exploit peculiarities of some AI systems, and achieve a significant speedup over traditional approaches by decreasing the combinatorial costs associated with searching large spaces.

We have established a clear and easy-to-check syntactic condition for detecting symmetry in non-deterministic planning domains, and developed techniques to break it by construction of a more abstract formulation whose solution can automatically aid in solving the original problem, providing, in particular, a radical reduction of the number of variables involved in making plans, and thereby giving *polynomial time* solutions to the original planning problems.

Our results are 'orthogonal' to traditional logical systems but are easily specified and handled in terms of linear logic, forming a bridge between *human common-sense reasoning* and problem solving, on one hand, and *computer-aided planning* and the ability of the automated systems to reason effectively in complex but natural domains, on the other hand.

From the methodological point of view, we have proposed an approach to planning which does not take the *unique names assumption* for granted — indeed, it shows how this *unique names assumption* can be a considerable hindrance — and shows, also, that there is a rigorous correspondence between proofs and plans (and thus that proof theory might be a more appropriate theoretical discipline for these problems than Tarski-style set-theoretical semantics).

Acknowledgments

We are greatly indebted to François Lévy for his fruitful discussions.

We owe special thanks to the referees for their insightful comments and fruitful recommendations, which have allowed to improve the exposition of this paper.

The work is partially supported by project GEOCAL ACI *Nouvelles interfaces des mathématiques*.

For further reading

[7,9,10,17,34].

References

[1] AIPS'98 Planning Systems Competition, http://www.informatik.uni-freiburg.de/~koehler/aips.html.
[2] AIPS-00 Planning Competition, http://www.cs.toronto.edu/aips2000/.
[3] W. Bibel, A deductive solution for plan generation, New Generation Computing 4 (1986) 115–132.
[4] C. Boutilier, T. Dean, S. Hanks, Decision-theoretic planning: Structural assumptions and computational leverage, Journal of AI Research (JAIR) 11 (1999) 1–94.
[5] C. Boutilier, T. Dean, S. Koenig, Editorial, Artificial Intelligence 147 (2003) 1–4.
[6] Tom Bylander, The Computational Complexity of Propositional STRIPS Planning, Artificial Intelligence 69 (1994) 165–204.
[7] Tom Bylander, A linear programming heuristic for optimal planning, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence, 1997, pp. 694–699.
[8] A. Cimatti, M. Pistore, M. Roveri, P. Traverso, Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence 147 (2003) 35–84.
[9] K. Erol, D. Nau, J. Hendler, A critical look at critics in HTN planning, in: IJCAI-95, Montreal, August 1995.
[10] K. Erol, D. Nau, V.S. Subrahmanian, Complexity, decidability and undecidability results for domain-independent planning, in: Planning, Artificial Intelligence 76 (1–2) (July 1995) (special issue on planning).
[11] D. McDermott, J. Hendler, Planning: What it is, What it could be, An introduction to the special issue on planning and scheduling, Artificial Intelligence 76 (1995) 1–16.
[12] M. Fox, D. Long, The detection and exploitation of symmetry in planning domains, in: Proc. IJCAI 2001, pp. 956–961.
[13] E. Guéré, R. Alami, One action is enough to plan, in: Proc. IJCAI 2001, pp. 439–444.
[14] H. Geffner, Functional strips: A more flexible language for planning and problem solving, in: Logic-based Artificial Intelligence, Kluwer International Series in Engineering and Computer Science, 2000, pp. 187–209.
[15] V. Gehlot, C.A. Gunter, Normal process representatives, in: Proc. 5th Annual IEEE Symposium on Logic in Computer Science, Philadelphia, June 1990.
[16] J.-Y. Girard, Linear logic, Theoretical Computer Science 50 (1) (1987) 1–102.
[17] J.-Y. Girard, Linear logic: Its syntax and semantics, in: J.-Y. Girard, Y. Lafont, L. Regnier (Eds.), Advances in Linear Logic, in: London Mathematical Society Lecture Notes, vol. 222, Cambridge University Press, 1995, pp. 1–42.
[18] P. Haslum, P. Jonsson, Some results on the complexity of planning with incomplete information, in: S. Biundo, M. Fox (Eds.), Recent Advances in AI Planning (ECP'99), in: Lecture Notes in Artificial Intelligence, vol. 1809, Springer-Verlag, pp. 308–318.
[19] M.I. Kanovich, Horn programming in linear logic is NP-complete, in: Proc. 7th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, June 1992, pp. 200–210.
[20] Max Kanovich, Linear logic as a logic of computations, Annals of Pure and Applied Logic 67 (1994) 183–212.
[21] M.I. Kanovich, The direct simulation of Minsky machines in linear logic, in: J.-Y. Girard, Y. Lafont, L. Regnier (Eds.), Advances in Linear Logic, in: London Mathematical Society Lecture Notes, vol. 222, 1995, pp. 123–145.
[22] M. Kanovich, J. Vauzeilles, The classical AI planning problems in the mirror of Horn linear logic: Semantics, expressibility, complexity, Journal of Mathematical Structures in Computer Science 11 (2001) 689–716.
[23] M. Kanovich, J. Vauzeilles, Coping polynomially with numerous but identical elements within planning problems, in: Proc. Annual Conference of the European Association for Computer Science Logic, CSL'03, Vienna, Austria, 25th–30th August 2003, in: Lecture Notes in Computer Science, vol. 2803, 2003, pp. 285–298.
[24] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: Proc. AAAI-1996, Portland, OR.
[25] A.P. Kopylov, Decidability of linear affine logic, Information and Computation 164 (1) (Jan 2001) 173–198.
[26] Y. Lafont, The finite model property for various fragments of linear logic, Journal of Symbolic Logic 62 (4) (1997) 1202–1208.
[27] M.L. Littman, J. Goldsmith, M. Mundhenk, The computational complexity of probabilistic planning, Journal on Artificial Intelligence Research 9 (1998) 1–36.
[28] M. Masseron, C. Tollu, J. Vauzeilles, Generating plans in linear logic: I. Actions as proofs, Theoretical Computer Science 113 (1993) 349–370.
[29] M. Masseron, Generating plans in linear logic: II. A geometry of action, Theoretical Computer Science 113 (1993) 371–375.
[30] N.J. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann, 1998.
[31] Proceedings of SAT-2002, Fifth International Symposium on the Theory and Applications of Satisfiability Testing, May 6–9, Cincinnati, OH, USA, 2002.

[32] F.P. Ramsey, On a problem of formal logic, Proceedings of London Mathematical Society 30 (series 2) (1930) 264–286.
[33] J. Rintanen, Symmetry reduction for SAT representations of transition systems, in: Proc. 13th International Conference on Automated Planning and Scheduling, AAAI Press, 2003, pp. 32–40.
[34] A. Scedrov, Linear logic and computation: A survey, in: H. Schwichtenberg (Ed.), Proof and Computation, Proc. Marktoberdorf Summer School 1993, NATO Advanced Science Institutes, Berlin, 1994, pp. 281–298.